3-23-2017

Optimal Design of a Hexakis Icosahedron Vacuum Based Lighter than Air Vehicle

Joseph R. Schwemmer

Optimal Design of a Hexakis Icosahedron Vacuum Based Lighter than Air Vehicle

THESIS

Joseph R. Schwemmer, 2d Lt, USAF

AFIT-ENS-MS-17-M-158

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENS-MS-17-M-158

OPTIMAL DESIGN OF A HEXAKIS ICOSAHEDRON VACUUM BASED LIGHTER THAN AIR VEHICLE

THESIS

Presented to the Faculty

Department of Operational Sciences

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Joseph R. Schwemmer, B.S.

2d Lt, USAF

March 2017

AFIT-ENS-MS-17-M-158

OPTIMAL DESIGN OF A HEXAKIS ICOSAHEDRON VACUUM BASED LIGHTER THAN AIR VEHICLE

THESIS

Joseph R. Schwemmer, B.S.
2d Lt, USAF

Committee Membership:

Dr. James W. Chrissis
Chair

Dr. Anthony Palazotto
Co-Chair

Dr. Carl Parson
Member

AFIT-ENS-MS-17-M-158

Abstract

Due to the rising cost and scarcity of helium, new methods to ensure buoyancy for lighter-than-air vehicles (LTAVs) are being sought. One alternative under study uses an internal vacuum to reduce the weight to buoyancy ratio. It's a novel approach; however, the vacuum presents challenges for the vehicle's structure. The structure must have minimum mass while preventing buckling and excess stress throughout the frame and membrane. The structure under analysis is a hexakis icosahedron with a membrane covering. Achieving minimum mass involves optimizing the structure under the loading conditions. Finite-element analysis (FEA) and direct-search meth-ods are employed, providing an optimal design under various regimes. Specifically, ABAQUS ◯ R is used as a FEA modeler, and mesh-adaptive direct search (MADS) is the optimization procedure. The goal of this research is to reduce the diameter of the vehicle using optimization techniques to a goal size of 31 inches (0.7874 meters). The smallest design to date has a diameter of 20 feet (6.096 meters). This research demonstrates the feasibility of two designs, one at 15 feet (4.572 meters) and another at 4 feet (1.2192 meters). The problem formulation includes multiple black-box objectives and constraints. Results for a number of designs are presented and compared.

Acknowledgements

I would like to thank my advisors, Dr. Chrissis and Dr. Palazotto, for their continuous support and time commitment in this research effort.

Additionally, Maj Todd Paciencia was instrumental in the modification and utilization of the MADS code implementation in MATLAB. Lt Col Chad Hale assisted in the use of the High Performance Computer at the Air Force Research Laboratory. Mike Snure from the Air Force Research Laboratory provided material properties for graphene and two metal aerogels. Without their efforts, this research would not have been possible.

Lastly, I would like to thank Capt William Caballero, CPT Alexander Kline, Capt Phillip Jenkins, and 2d Lt Hai-Dang Nguyen for their help in editing my thesis document and providing feedback.

Joseph R. Schwemmer

Table of Contents

Page

List of Figures

List of Tables

OPTIMAL DESIGN OF A HEXAKIS ICOSAHEDRON VACUUM BASED LIGHTER THAN AIR VEHICLE

I. Introduction

Structural optimization enhances the efficacy of a system by altering the size, shape, and topology of the structure of an object under development. It is a tool utilized by engineers and scientists to produce an object with certain characteristics, such as minimal mass, minimized deflection, or other such design properties. Structural optimization utilizes a variety of optimization methods, including mathematical programming and evolutionary methods, to determine the optimal thickness and locations of members to support loads at fixed points [2]. Analysis of complex structures is particularly difficult because of corresponding complexity in the mathematical models used in the analysis and optimization, often due to object size, complexity, granularity of nodes, or complex force distributions. To overcome these difficulties, finite-element analysis (FEA) or computational fluid dynamics (CFD) combined with optimization algorithms are often utilized.

The optimization methods employed in structural design optimization are too numerous to list exhaustively, but include gradient-based methods, evolutionary methods, and direct search. As an example of similar studies, in 2012 Gern produced an analysis and optimization of a Hybrid Wing Body (HWB) design based on FEA producing a variety of vehicle class sizes using the Nastran SOL200 suite [3]. Another approach integrating FEA as the underlying modeler and direct search as the optimizer was carried out by Parson [4]. Both of these cases begin with a FEA model, with Parson using the computer-aided design (CAD) program ABAQUS ®, to indicate

the nodes and forces on the item under design. The design variables and constraints are defined by stresses, forces, deflections, and specific instance characteristics. The underlying design is then optimized for the objective, usually minimum mass, using the direct search algorithm, MADS (Mesh Adaptive Direct Search) developed by Audet and Dennis [5], and modified and enhanced by others. This thesis applies some of the techniques explored by Parson on a unique structure, the *hexakis icosahedron*.

The specific instance of structural optimization examined is to minimize mass on a hexakis icosahedron shaped vacuum lighter-than-air vehicle (LTAV). LTAVs maintain lift through a buoyant force, commonly using an internal gas that is less dense than air, such as helium or hydrogen, to displace the heavier air. The air displaced by the vehicle must weigh more than the vehicle itself for positive buoyancy. A vacuum could be used as a replacement to the internal gas, to provide a structure that displaces an airmass greater than its weight. An internal vacuum is able to provide lift because it has no mass and the containment structure displaces more air than it weighs. The weight of a vacuum LTAV derives from the structure needed to maintain a vacuum in a certain shape. The structure must be rigid to resist the internal forces yet light enough to be positively buoyant. The driving factor behind vacuum LTAVs is the decreasing availability of helium [6] and the diminishing stores of helium in United States possession [1]. In addition, the Helium Stewardship Act of 2013 was passed to help stabilize the market for helium [7] by conserving its use. Thus, an alternate lifting force for LTAVs would correspondingly reduce demand for the gas. A hexakis icosahedron is shown in Figure 1. This example structure has 62 vertices, 180 edges, and consists of 120 scalene triangle faces.

The first reference to vacuum LTAVs was by Francesco Lana de Terzi in 1663 [8], investigated further by Akhmeteli and Gavrilin [9]. De Terzi's design used a copper sphere. Unfortunately, such a design was not feasible with materials available at that

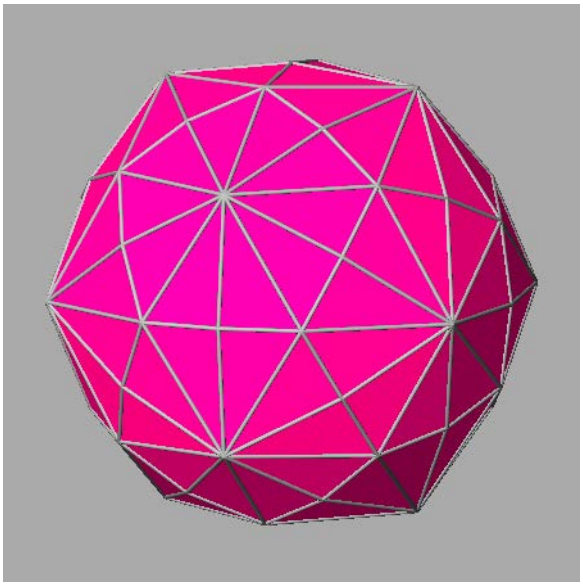


Figure 1. Hexakis Icosahedron [1]

time due to shell buckling and weight [1]. Even the best materials today still do not attain the required specific modulus, also called specific stiffness, required to prevent shell buckling for the sphere, which is estimated at 4.9×10^8 $m^2 s^{-2}$ [1]. Figure 2 shows the de Terzi design.

In 2005, Akhmeteli and Gavrilin developed a layered shell vacuum LTAV that would prevent shell buckling [9]. However, this design cannot be manufactured with current technology. Spheres are ideal for LTAVs because they maximize the amount of internal volume for a given surface area. Instead of using a sphere, a near-sphere geodesic shape could be used to provide a stronger structure. A design by Metlen in 2012 constructed the icosahedron design using cylindrical rods and a membrane skin [10]. Using this initial design, Cranston developed a hexakis icosahedron to achieve more robust weight-to-buoyant-force ratios while maintaining feasibility [1].

The primary attribute when investigating the structure of LTAVs is the *weight-to-buoyancy ratio*. A value equal to 1 indicates a neutrally buoyant object, values greater than 1 indicating negative buoyancy, and less than 1 for positive buoyancy. Figure 3 illustrates this buoyancy behavior. Investigations by Cranston demonstrated that the

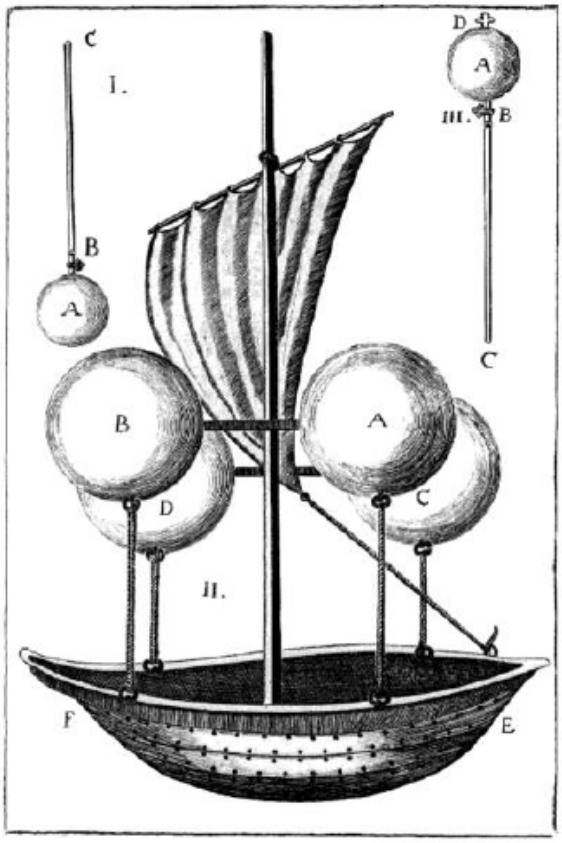


Figure 2. Lana de Terzi's Vacuum LTAV [1]

icosahedron design would not achieve the required weight-to-buoyancy ratio for the payload and altitude desired for the vehicle. He shows that with current materials the icosahedron design is not feasible due to altitude, c-ratio (beam thickness-to-radius ratio), and the skin weight-to-buoyancy ratio. Feasible designs were found using a radius in the range of 2.5 to 6 meters [1].

The method implemented to perform this task takes the forces applied to the vehicle through ABAQUS ® to solve for the stresses and deflections at various points of the structure. These results are used to compute the weight, buoyancy, maximum stress on the membrane/frame, and maximum deflections of the membrane/frame. The maximum stresses, deflections, and weights are used in an optimization method that is iteratively driven by the Mesh Adaptive Direct Search (MADS) algorithm,

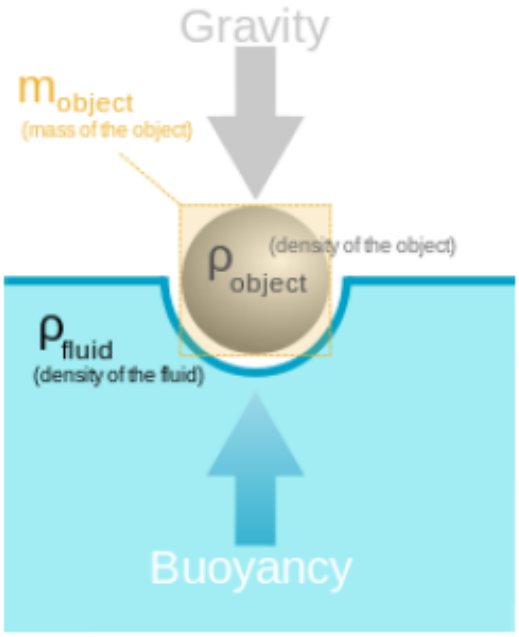


Figure 3. Buoyancy Diagram [1]

with the objective to minimize mass. The process is iterative, where the results from MADS and ABAQUS ® are shared until a final design is reached through convergence to an indicated minimum mass (i.e., within a specified tolerance). A more complete description of MADS is given in Chapter II. The generic objective function for this type of optimization problem is typically defined as

$$\arg \min_{\omega \in \Omega} F = f(\omega) \tag{1}$$

with Ω as the design space and ω as a specific design vector [11]. Equation 1 states that a function of the design vector should be minimized while ensuring the design vector is in the design space, meaning that it is feasible with respect to all constraints.

As with any modeling analysis, some assumptions are made. The forces are assumed to be sea-level pressures, because that is a practical force level a lighter-than-air vehicle will encounter. If the vehicle does not float at sea-level, it will not float at

other altitudes because the air density, and therefore the buoyancy of the structure, decreases as altitude increases. The relationship between altitude, structure size, and payload can be defined as

$$\text{altitude} \propto \frac{\text{structure size}}{\text{payload}}. \tag{2}$$

The payload is defined as any mass not related to the structure of the vehicle. As altitude increases, the air density decreases. Therefore, the size of the structure would need to increase to maintain the same payload capacity. Any design must be able to float at sea level if it is to float at any altitude, given the relationship in equation 2. The objective does not have a readily available derivative because it is produced by the finite-element analysis; thus, a derivative-free optimization methodology, such as MADS, is appropriate. The only forces modeled on the vehicle are the vacuum-induced loads because they are the main concern for producing feasible LTAVs. Any environmental forces on the vehicle are not examined. Using the MADS optimization tool and ABAQUS ® FEA solver, the minimum-mass structure of a hexakis icosahedron vacuum LTAV is produced.

Two original problem formulations are developed. One formulation changes the beam thickness, beam radius, and skin thickness to produce a minimum mass and minimum deflection vehicle. The other formulation investigates the material densities required to achieve a floating vehicle of varying diameters. Constraints on the manufacturing limitations, yield stresses, and ratio of beam thickness to beam radius are included in both formulations.

These problem formulations transition away from current research in the vacuum LTAVs. Previous and current research in these structures sets the weight-to-buoyancy ratio first and finds the beam and membrane geometries for the given ratio. This methodology creates infeasible designs due to manufacturing limitations. The

methodology implemented in this research ensures all designs can be manufactured.

The subsequent chapters of this thesis are organized as follows. First, relevant research in the areas of optimal design, mathematical optimization methods, derivative-free optimization, and previous work in design and analysis of hexakis icosahedrons are discussed in Chapter II. Methods of optimization and design, as they specifically apply to this research, are presented in Chapter III. Finally, the results of this analysis and conclusions, along with recommendations for further work, are given in Chapter IV.

II. Literature Review

The work presented in this thesis is the optimization of the design of a new structure for a lighter-than-air vehicle (LTAV). Due to the complex nature of the structure, a number of disciplines are involved in the execution of this work. This chapter surveys the relevant literature of the areas required to carry out this research. Many of these disciplines are quite broad; in no way is this literature review intended to be exhaustive. Key references are cited as required, and others are included in the bibliography for completeness.

2.1 Structural Design Optimization

Structural Optimization involves optimization of designs in solid (*i.e.*, structural) or fluid mechanics. Frequently the objective is to determine the minimum-weight design for a particular type of structure subject to various manufacturing, design, operational, and environmental constraints [12]. Problems of this nature, where the components defining the optimization problem come from multiple disciplines, are often called *multi-disciplinary optimization* problems [13] [14]. The amount of time and effort required for solving these types of problems depends heavily on the complexity of the problem at hand, physically as well as mathematically.

In some cases, the design of an optimal structure can be solved as a linear program. The details for such an approach are given by Vanderbei [2], as well as others. Vanderbei shows that the structural design problem may be solved as a network-type problem. The graph of the problem instance shows the nodes as the joints of the structure with the arcs between nodes representing the members (beams). An example structural design problem topology is shown in Figure 4. The forces on the object are used to provide constraints for the underlying optimization problem, as the forces

must balance for a static item. The formulation for a minimum-weight structural design problem becomes

$$\text{minimize} \sum_{\{i,j\}\in A} l_{ij}|x_{ij}|$$

$$\text{subject to:} \sum_{j:\{i,j\}\in A} u_{ij}x_{ij} = -b_i,\ i = 1, 2, \ldots, m.$$

The decision variables x_{ij} represent the force exerted on the points i and j by the beam. The b_i terms are the external loads applied to each node i. u_{ij} represents the unit vector along each beam i, j with l_{ij} representing the length of the member. Vanderbei assumes that the weight of the beam is directly proportional to its volume and the density for each member is identical [2]. These assumptions are realistic in that the density of a material does not change by a significant amount in each beam, and beam weight increases as its size increases.

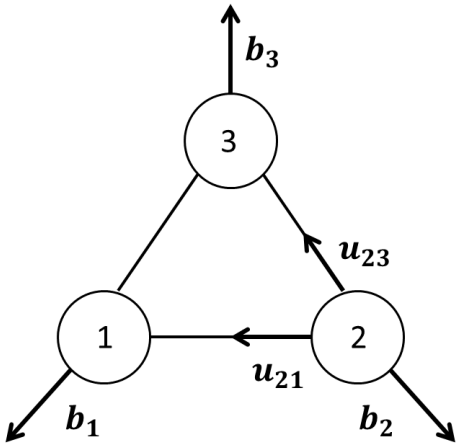


Figure 4. Example 2D Structural Design Problem

A true linear program is obtained by setting $x_{ij} = x_{ij}^+ - x_{ij}^-$, where $x_{ij}^+, x_{ij}^- \geq 0$. x_{ij}^+ is the tension force on the beam i, j and x_{ij}^- is the compression force. This substitution

results in removal of the absolute value found in the objective function.

Vanderbei's formulation does not account for structural deflection at the nodes, which could have a significant effect on a LTAV. These vehicles rely on maintaining a certain volume to provide sufficient buoyant force. Therefore, any deflection of a node could reduce the volume of the structure and ground the LTAV. Structures built with non-metals, which includes most LTAVs, usually exhibit nonlinear behavior because of structural deflection. Since this linear formulation does not account for nonlinear effects, this type of optimization approach cannot be used for the hexakis icosahedron. Thus, some other approach which accommodates the nonlinear concerns, the number of members used in the structure, and the force distribution, must be selected. The structural designs presented by Vanderbei are typically two-dimensional because a three-dimensional object has a more complex force distribution. The loads presented by the vacuum will not appear just at nodes, but rather carry across the entire surface. Additionally, this linear approach to design optimization does not account for a membrane covering a frame. The use of a membrane for containment changes how each force applies to each node; forces are present along the beam, not just concentrated at the nodes. The sections that follow address the selection of appropriate methodologies for design of a hexakis icosahedron structure having optimal performance characteristics.

2.2 Optimization Methods

Generally speaking, the field of optimization/mathematical programming is broad and varied. Application areas include problems as diverse as vehicle routing, capital budgeting, and network optimization problems. Texts such as Winston [15] and Hillier and Lieberman [16] cover many of these areas in a broad sense. More detailed theoretical and algorithmic coverage is given in Bazaraa, Sherali and Shetty [17],

and in Bertsekas [18]. The text "Engineering Optimization:Theory and Practice" by Rao [19] is a comprehensive, for the time it was written, presentation of the theory and methods applicable to broad engineering problems, with an engineering-oriented presentation. The entire field of optimization/mathematical programming is not surveyed here; rather some methods and techniques that specifically apply to structural design optimization problems are discussed.

Structural Optimization.

The structural design optimization problem involves establishing the "best" design when considering size, shape, or topology of the structure under analysis. The objective may be to minimize mass or perhaps minimize displacement of some member. The textbook by Arora, "Introduction to Optimum Design" [20], has been used for many years as a fundamental introduction to the field of design optimization. Material in that text describes the overall design process, and presents various methods and techniques that are used to bring about the optimal design.

Various optimization methods have been applied to structural design problems. The particular method generally depends on the the type of structure and its complexity, the complexity of the underlying mathematical model, the type of approximation method (FEA, CFD, etc.), and the preference of the designer. That is, some designers prefer derivative-based methods, some evolutionary methods, and others optimum-seeking direct search.

As previously alluded, the design and optimization of structures frequently involves solving problems modeled using nonlinear optimization models, for the objective and possibly the constraint(s). Thus, many of the algorithms applied to structural optimization problems are from the field of nonlinear programming. The "no free lunch (NFL)" theorem (in search and optimization) of Wolpert and Macready,

states, in loose terms, that no algorithm works equally well on all problems. This is evidenced by the number of methods which exist for solving different classes of optimization problems [21].

Certain optimization methods have been successfully applied to large-scale structural design problems. SQP (Successive Quadratic Programming) and SCP (Successive Convex Programming) are two related methods that have been applied to a variety of such structural design problems. Schittkowski, in collaboration with others, has published widely in these areas [22] [23]. These SQP and SCP analyses have included models where finite elements were used to approximate the underlying topology of the structure. Abramson, in his 1994 thesis and subsequent journal article [24] [25], used SQP with active-set modifications in the ASTROS environment to solve structural design test problems significant for the time, giving better solutions in less computational time than previously reported. Sriver, in his thesis, used SCP to solve a similar suite of test problems to optimality for the first time [26].

SQP and SCP require derivative information, either analytically-derived, or approximated numerically. This is considered a disadvantage of these methods, although the derivatives do provide useful information. Due to the complex nature of the structural design problem, derivative information is typically not available. Thus, derivative-free methods are often employed [11]. Some of these methods are evolutionary, such as using a genetic algorithm [27] [28]. Other researchers have solved these design problems by employing simulated annealing [29]. Derivative free optimization is ideal for a structural design problem because the objective function is too complex for many traditional methods.

2.3 Direct Search

A main class of derivative-free optimization methods is *direct search*. One type of direct search was presented by Hooke and Jeeves as a method to solve problems faster than classical methods, considering a different set of assumptions that admitted more types of functions (discrete, discontinuous, non-smooth, and others) [30]. Direct search allows the user to alter problem formulations without the process of tuning parameters as featured in evolutionary and heuristic methods. Direct search samples the objective function at a finite number of points during each iteration and decides where to move next based only on those objective function evaluations and a given search strategy [11].

The basic direct search algorithm updates the decision variable vector using

$$x_{k+1} = x_k + \alpha_k s_k$$

with α as the step size and s as the search direction. This method of searching the solution space for the optimal design vector is extended by Audet. Audet [31] provides the general algorithm for direct search: initialization, search and poll, and parameter update. Similar to other search algorithms, an initialization step is required to set algorithm tolerances and parameters and provide the choice of an initial solution vector. The polling step involves local searches in multiple directions off the current solution vector (*e.g.* local exploration). This process is performed iteratively to produce a final solution vector that satisfies the constraints and optimizes the specified objective function.

The search step evaluates the function at a set number of points that can be user-defined. This step is not required for convergence properties, but can be used practically to speed up the convergence of the algorithm [11]. The poll step is only

used if the search step is unsuccessful; it evaluates the function at points in the poll set. The poll set is defined in Algorithm 1. The poll step is used to ensure the objective function improves (decreases) and the algorithm is converging. A description for a directional direct-search method is shown in Algorithm 1 [11]. The poll step performs at most $|D_k|$ function evaluations and the stopping criteria is based on some α_{tol} for a chosen toleration in the step-size parameter.

Algorithm 1 Directional Direct-Search Method

Initialization: Choose $x_0, \alpha_0 > 0, 0 < \beta_1 \leq \beta_2 < 1$, and $\gamma \geq 1$. Let D be a set of positive bases.

for $k = 0, 1, 2, \ldots$ **do**

Search Step: Try to compute a point with $f(x) < f(x_k)$ by evaluating the function f at a finite number of points. If such a point is found, then set $x_{k+1} = x$, declare the iteration and the search step successful, and skip the poll step.

Poll Step: Choose a positive basis D_k from the set D. Order the poll step $P_k = \{x_k + \alpha_k d : d \in D_k\}$. Start evaluating f at the poll points following the chosen order. If a poll point $x_k + \alpha_k d_k$ is found such that $f(x_k + \alpha_k d_k) < f(x_k)$, then stop polling, set $x_{k+1} = x_k + \alpha_k d_k$, and declare the iteration and the poll step successful. Otherwise, declare the iteration (and the poll step) unsuccessful and set $x_{k+1} = x_k$.

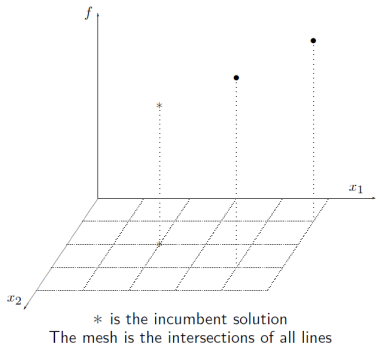
Mesh Parameter Update: If the iteration was successful, then maintain or increase the step size parameter: $\alpha_{k+1} \in [\alpha_k, \gamma\alpha_k]$. Otherwise, decrease the step size parameter: $\alpha_{k+1} \in [\beta_1\alpha_k, \beta_2\alpha_k]$.

end for

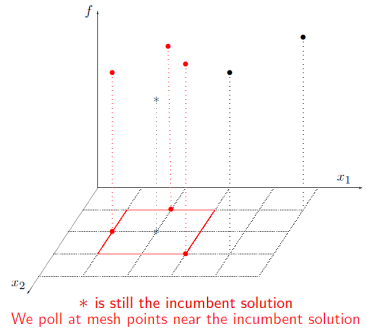
The step size parameter, α, converges to zero as the optimization progresses and the vector x converges to the global optimum. These results are guaranteed with the positive basis D_k and if the function has some differential properties [11].

An illustration of Direct Search is shown in Figure 5. Figure 5a shows the initial solution and two search points. The search points do not indicate improvement in the objective function value, so the poll step is conducted. The poll step is shown in Figure 5b. The poll points are chosen using a set step size and a vector from the positive basis. None of the poll points improve the objective function, so the step size parameter is decreased to bring the poll points closer to the current best solution,

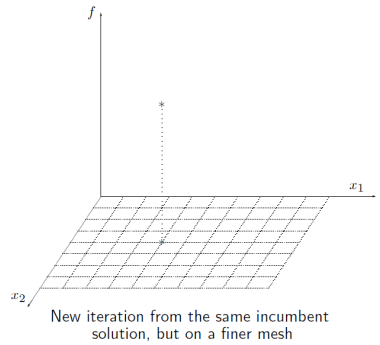
shown in Figure 5c. If the iteration is successful at finding a better solution, the step size stays the same or increases to search a broader area.



(a) Initialize and Search



(b) Poll



(c) Refine Mesh

Figure 5. Visualization of Direct Search/MADS

An example problem using this algorithm is now demonstrated. Let the problem be defined as

$$\text{minimize } f(\mathbf{x}) = (x_1 - x_2 - 2)^2 + 3x_2 - 5$$

subject to:

$$x_1 \geq 0$$

$$x_2 \geq 0.$$

First, choose $\mathbf{x}_0, \alpha_0 > 0, 0 < \beta_1 \leq \beta_2 < 1$, and $\gamma \geq 1$ as described in Algorithm 1. Let $\mathbf{x}_0 = (0,0), \alpha_0 = 1, \beta_1 = 0.1, \beta_2 = 0.5$ and $\gamma = 2$. The initial solution evaluates as $f(\mathbf{x}_0) = -1$. Next, execute two search points at $\mathbf{x}_0^{(1)} = (1,0)$ and $\mathbf{x}_0^{(2)} = (0,1)$. $f(\mathbf{x}_0^{(1)}) = -4$ and $f(\mathbf{x}_0^{(2)}) = 7$, so this search step is successful. The current solution is at $\mathbf{x}_1 = \mathbf{x}_0^{(1)}$. Now increase the step-size parameter to $\alpha_1 = 2$ using $\gamma = 2$. Proceed to the poll step for the next iteration. Let the positive basis be

$$\mathrm{D} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This basis will be used for all poll steps. Order the poll step, $\mathbf{x}_1^{(1)} = (1+2*1, 0+2*0) = (3,0)$ and $\mathbf{x}_1^{(2)} = (1+2*0, 0+2*1) = (1,2)$. $f(\mathbf{x}_1^{(1)}) = -4$ and $f(\mathbf{x}_1^{(2)}) = 10$, therefore stop polling and declare this iteration unsuccessful. The iteration is unsuccessful because the objective value has not been improved (decreased). The current solution is $\mathbf{x}_1 = \mathbf{x}_0^{(1)}$.

Decrease the step-size parameter using $\beta_2 = 0.5$ giving the new step size parameter $\alpha_2 = 1$. The next poll step begins, with $\mathbf{x}_1^{(3)} = (2,0)$ and $\mathbf{x}_1^{(4)} = (1,1)$. $f(\mathbf{x}_1^{(3)}) = -5$, so this iteration is successful because an improved objective value has been found and $\mathbf{x}_1^{(4)}$ is not evaluated. The step size parameter is increased to $\alpha_3 = 2$ with the current

solution as $\mathbf{x}_2 = \mathbf{x}_1^{(3)}$. The next two poll points are $\mathbf{x}_2^{(1)} = (4, 0)$ and $\mathbf{x}_2^{(2)} = (2, 2)$. $f(\mathbf{x}_2^{(1)}) = -1$ and $f(\mathbf{x}_2^{(2)}) = 5$, therefore this iteration is unsuccessful and the step-size parameter is reduced to $\alpha_4 = 0.2$ using $\beta_1 = 0.1$.

The next iteration's poll steps are $\mathbf{x}_2^{(3)}$ and $\mathbf{x}_2^{(4)}$ with locations $(2.2, 0)$ and $(2, 0.2)$. The objective function values are -4.69 and -4.36 respectively for these poll points. This iteration is unsuccessful, so the step size parameter is reduced again to $\alpha_5 = 0.02$. This parameter is approaching zero, and the global minimum will be found at $(2, 0)$, as can be seen by inspection of the objective function. A summary of the steps and results for this example are shown in Table 1.

Table 1. Direct Search Example Problem Steps and Results

Step	**Step Type**	**x**	**f(x)**
0	Initial	$(0, 0)$	-1
1	Search	$(1, 0)$	-4
2	Search	$(0, 1)$	7
3	Poll	$(3, 0)$	-4
4	Poll	$(1, 2)$	10
5	Poll	$(2, 0)$	-5
7	Poll	$(4, 0)$	-1
8	Poll	$(2, 2)$	5
9	Poll	$(2.2, 0)$	-4.69
10	Poll	$(2, 0.2)$	-4.36

A special instance of direct search, namely Mesh Adaptive Direct Search (MADS) is presented in the following section and is the algorithm implemented for this research.

Mesh Adaptive Direct Search.

In 2012, Parson optimized a flapping-wing structure using a FEA representation of the structure, and Mesh Adaptive Direct Search (MADS) as the optimization engine [4]. The results of the optimization analysis provided a structure that maintained certain structural and performance characteristics, while minimizing mass. The work

in this thesis follows a similar direction, but with a completely different structural configuration. The following section details the MADS methodology.

The method selected for use in this research is the MADS methodology. *Mesh Adaptive Direct Search* (MADS), developed by Audet and Dennis, extends the analytic logic of the fundamental Generalized Pattern Search methodology to include search over a specified search mesh [5]. That is, MADS introduces a search mesh into Algorithm 1 where the mesh is defined as

$$M_k = \left\{ x_k + \alpha_k D u : u \in \mathbb{Z}_+^{|D|} \right\} \tag{3}$$

where D is allowed to be an infinite set of positive bases, but in practice is a finite set. Additionally, MADS forces the search and poll steps from Algorithm 1 to only evaluate points in M_k. The step size parameter is updated using the following procedure: Choose a rational number $\tau > 1$, a nonnegative integer $m^+ \geq 0$, and a negative integer $m^- \leq -1$. If the iteration is successful, the step size parameter is updated by $\alpha_{k+1} = \tau^{m_k^+} \alpha_k$ with $m_k^+ \in \{0, \ldots, m^+\}$. Otherwise, the parameter updates by $\alpha_{k+1} = \tau^{m_k^-} \alpha_k$ with $m_k^- \in \{m^-, \ldots, -1\}$. The step size parameter rules above match the rules of Algorithm 1 as $\beta_1 = \tau^{m_k^-}, \beta_2 = \tau^{-1}$, and $\gamma = \tau^{m_k^+}$. This procedure ensures strong convergence properties for the MADS approach.

MADS is capable of achieving global convergence for nonsmooth functions provided the function is Lipschitz continuous near the optimal solution [11] [5]. As the algorithm progresses, the poll points eventually fail to find improving solutions and the mesh is refined. As the mesh refines, the step size α converges to zero. The limit point of this sequence is the solution vector x^*, called the refining direction for x. This limit point has $f^o(x^*; v) \geq 0$ for all refining directions v [11]. The special function used here is a generalized directional derivative, where the Clarke calculus is used

to produce these generalized directional derivatives. They approximate the function partial derivatives, the gradient, at a point [32]. With the directional derivatives all positive, the function must be at the global minimum.

The convergence of MADS has been discussed by Torczon and Abramson. Torczon describes that for nonlinear programming, global convergence means that the solution found is optimal at the first-order from any starting point. She continues that it is difficult to prove second-order convergence results without a derivative structure, but algorithms usually find good local minimums [33]. However, Abramson and Audet were able to prove that MADS converges to second-order stationary points [34]. The convergence proof depends on using an increasing number of poll directions for each iteration, which is not practical. To remedy this, specific implementations of MADS are used, such as LTMADS or OrthoMADS, that choose a subset of the poll directions in each iteration. General convergence is not guaranteed but the algorithm converges in the long run with probability 1 [34]. Abramson and Audet do state that although this convergence property is weaker, it still works in practice.

MADS was further investigated by Abramson, who demonstrated the algorithm could be formulated for mixed variable problems [35]. Abramson also developed an instance of MADS that uses orthogonal polling directions chosen deterministically [36]. OrthoMADS was used as the MADS instance for this research. OrthoMADS chooses the poll directions without uncertainty (as opposed to LTMADS) and the poll directions are orthogonal to each other, as the name infers [36]. The concepts of MADS have been extended to handle multi-objective formulations using a Pareto front [37], as well as stochastic cases [38]. Surrogate and sampling strategies have also been added to the basic MADS code [39].

Some applications addressed using MADS include optimizing the number and placement of injectors in a scramjet fuel injection array [40] and structure determi-

nation of nanomaterials [41]. The applications of the algorithm are varied due to its generality. Unlike a typical heuristic algorithm, MADS is provably convergent and does not have parameters that must be adjusted for a particular problem type. In addition, MADS is able to optimize multi-objective problems by considering a series of single-objective problems and generating a Pareto front [37]. The Pareto front is used to find the best solution possible given multiple objectives by identifying dominant solutions. Because of its convergence properties and ability to handle derivative-free results derived from computer simulations, MADS is the algorithm of choice used in this research to optimize the hexakis icosahedron.

2.4 Previous Work on Vacuum LTAVs

With the optimization algorithm decided for this research, the remainder of the literature review is a summary of prior work done addressing vacuum LTAVs. The initial study was conducted by Metlen, who investigated the feasibility of vacuum LTAVs as an alternative to traditional airframes [10]. He determined that a geodesic shape is required to achieve the desired weight-to-buoyancy ratio and looked into using an icosahedron. Adorno-Rodriguez started with the icosahedron and evaluated the design further. He investigated the size and shape of the beams and characterized the benefits of the membrane and material properties [42].

Cranston used design of experiments (DOE) techniques to investigate possible optimal designs of icosahedron and hexakis icosahedron vehicles. As stated in Chapter I, Cranston expanded upon the icosahedron design proposed by Metlen. He shows that the generic icosahedron was not feasible for a vacuum lighter-than-air vehicle (VLTAV) but a hexakis icosahedron has the weight-buoyancy ratio required. Cranston was able to produce two designs, one optimized for maximum payload and one for minimum vehicle radius [1]. These designs have diameters in excess of 20 feet (6.096

meters).

The primary measure of a LTAV is the weight-to-buoyancy ratio. The weight-to-buoyancy ratio is calculated as shown in Equation 4. V_s is the volume the skin, V_f is the volume of the frame, and ρ_s and ρ_f are the density of the skin and frame, respectively. V_i is the initial volume of the vehicle with V_r as the reduced volume with the vacuum load.

$$\frac{W}{B} = \frac{V_s\rho_s + V_f\rho_f + (V_i - V_r)\rho_{\text{air, i}}}{(V_i - V_r)\rho_{\text{air, o}}} \tag{4}$$

Another measure utilized in this research is the c-ratio, which is a ratio of the beam thickness to the beam radius. Cranston states that the c-ratio must be greater than 0.02 to prevent local buckling [1]. The c-ratio is best when its value is minimal because the moment of inertia for the beam increases as the thickness decreases. The two boundary conditions used for the structure to allow the FEA solver to complete its analysis are forcing two opposite nodes, top and bottom, to have zero displacement. These conditions force the structure to have symmetrical stress and displacement, which is representative of what the structure would encounter in use. Figure 6 shows this boundary condition.

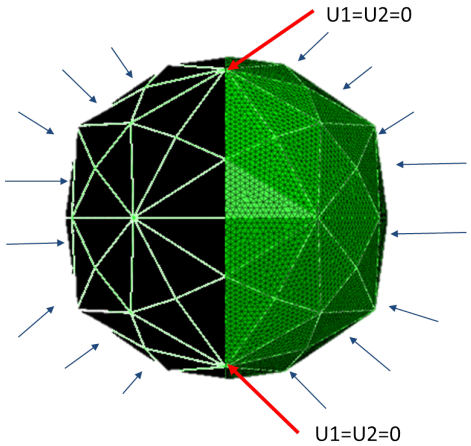


Figure 6. Boundary Conditions [1]

2.5 Summary

Due to the complexity of the structure, and the lack of readily-available derivative information, the use of a derivative-free optimization method is indicated. Previous use of the MADS methodology on problems of this nature further reinforced the decision to use this approach for the optimization engine. FEA, employed through ABAQUS ®, is used to analyze the structure and provide accurate stress and deflection information, leading to establishing a minimum-mass design of the hexakis icosahedron LTAV.

In Chapter III, the specifics of the hexakis icosahedron and the MADS algorithm as related to this specific design problem are discussed. The problem formulations, MADS coding, and ABAQUS ® calls are detailed.

III. Methodology

This chapter presents the algorithms used in this analysis, including the specific coding and modifications required to optimally design a hexakis icosahedron, subject to various operating conditions. It builds on the methods suggested from Chapter II and details the development of the FEA models, as well as the optimization as implemented via MADS.

3.1 Problem Formulations

Two types of problems were run using the MADS code. One type of problem focuses on optimizing the beam and skin geometry of the hexakis (referred to as *Problem Type One*), while the second problem type considers the material densities required to achieve a floating hexakis given a specified radius (referred to as *Problem Type Two*). For Problem Type One, the formulation for the hexakis icosahedron for an optimal structure given material properties, diameter, and altitude can be expressed as follows. Let x_1 = beam radius, x_2 = beam thickness, and x_3 = skin thickness, all

in meters. The problem formulation for the hexakis icosahedron is then:

$$\text{minimize } f\left(\frac{\text{Weight}}{\text{Buoyancy}}, \text{ Max Frame Displacement, Max Skin Displacement}\right)$$

subject to:

$$\frac{\text{Max Skin Stress}}{\text{Skin Yield Stress}} \leq 1$$

$$\frac{\text{Max Frame Stress}}{\text{Frame Yield Stress}} \leq 1$$

$$0.02x_1 - x_2 \leq 0$$

$$-0.025x_1 + x_2 \leq 0$$

$$x_1 \geq 0.0002 \text{ meters}$$

$$x_2 \geq 0.0002 \text{ meters}$$

$$x_3 \geq 0.0002 \text{ meters}$$

The first two constraints are non-linear and handle the stress limits of the structure. Von Mises stress is used to handle the principal stress loads and account for the greatest stress at a node. The next two conditions are based on the c-ratio (beam thickness/beam radius) values stated by Cranston [1]. This ensures the beam has a sufficient bending moment of inertia and avoids a region of local buckling. The c-ratio must be in the interval $[0.02, 0.025]$. The last three constraints are manufacturing limitations [1], which prevent the optimal solution from including material that cannot be manufactured. The manufacturing process has a resolution limit that bounds the size of the beam to be produced.

Three objectives are used to meet the requirements for the vacuum LTAV. The structure must have minimum mass to achieve a desired weight-to-buoyancy ratio. Also, the structure must be compliant, represented as a function of the displacement of the frame and the membrane. The displacement objectives are not considered if

the design does not float. These objectives are functions of the decision variables but the functions themselves are treated as "black-box". The stress and deflection of the structure is computed in ABAQUS ® and these values form the objective function values for deflection, the maximum stress on the frame and membrane for the non-linear constraints, as well as determining the final volume of the vehicle to solve for buoyancy.

The materials used for the frame and the membrane are held constant for this problem formulation. A carbon nanotube composite (CNT) is used as the frame and Spectra 1000 is used for the membrane. The properties for these materials are shown in Table 2 [42].

Table 2. Problem Formulation Type One Material Properties

Material	**Density (kg/m^3)**	**Poisson's Ratio**	**Modulus of Elasticity (GPa)**	**Yield Stress (GPa)**
CNT	1250	0.33	293	3.8
Spectra	970	0.33	172	3.0

The second type of problem formulation uses a given altitude, radius, beam structure, and some material properties to optimize the structure by maximizing material density. This problem type is designed to determine the material densities required to create certain vehicle sizes under current manufacturing limitations. The set material properties used for the skin and frame are:

Table 3. Problem Formulation Type Two Material Properties

Poisson's Ratio	**Modulus of Elasticity (GPa)**	**Yield Stress (GPa)**
0.33	100	3.0

Let x_1 = Frame Density and x_2 = Skin Density with the densities in kg/m^3. The formulation for the problem is then:

$$\text{maximize } f(x_1,\ x_2)$$

subject to:

$$\frac{\text{Max Skin Stress}}{\text{Skin Yield Stress}} \leq 1$$

$$\frac{\text{Max Frame Stress}}{\text{Frame Yield Stress}} \leq 1$$

$$\frac{\text{Weight}}{\text{Buoyancy}} \leq 1$$

$$x_1 - x_2 \leq 10$$

$$x_1 - x_2 \geq -10$$

$$10 \leq x_1 \leq 1000$$

$$10 \leq x_2 \leq 1000$$

As in the first problem formulation, the stress constraints are present. The primary objective of the first problem formulation, minimizing weight divided by buoyancy, becomes a nonlinear constraint in this problem to force designs that float. The linear constraints are present to ensure that the skin and frame density have a maximum difference of 10 kg/m^3. This prevents the optimizer from forcing the frame density to near zero because the frame composes at least 70% of the total weight of the structure with designs under 20 feet (6.096 meters) in diameter. The densities are allowed to vary between 10 and 1000 kg/m^3.

The manufacturing limitations and c-ratio constraint from Problem Type One are implemented in this formulation by setting the beam radius, beam thickness, and skin thickness of the vehicle at the start of the optimization. With small diameter designs, the absolute manufacturing limits with the c-ratio considerations are implemented.

These limits are a beam radius of 8 millimeters, a beam thickness of 0.2 millimeters, and a skin thickness of 0.2 millimeters. Increases in diameter also forced these values to increase. These values are shown in Table 4 in Chapter IV. Without increasing the beam radius, beam thickness, and skin thickness, the FEA does not converge to a solution.

The constraint set does not directly account for the beam brittleness and modulus of elasticity in either formulation. These material properties are incorporated into the FEA model. The feasibility of the structure due to deflection is also left for the FEA model to determine.

3.2 MADS Code

The specifics of the Mesh Adaptive Direct Search (MADS) algorithm are now discussed. As mentioned in Algorithm 1, certain parameters are needed to run MADS. These parameters increase or decrease the size of the mesh based on the movement toward the solution. For this research, the refining factor is 0.5 and the coarsening factor is 2.0 for the mesh update parameters. This means when a better solution is found in the search or poll steps, the mesh doubles in size and when a better solution does not exist, the mesh is reduced by half.

MADS is based off the direct search algorithm, therefore stopping criteria are required to terminate the process. One criterion is an "indifference zone" on the objective space; this defines a kind of sensitivity to precision in the solution values. The indifference zone for the objective values is the range of the objective space divided by 10 for each objective. A larger zone enables faster convergence, but may fail to find good solutions. The second criterion is a limit on the number of search and poll points. Five points are used in the search step and forty points are used in the poll step. These values are driven by the time required for one iteration of the

solver, approximately 15 minutes for each step.

Direct search methods require an initial solution. An initial solution is chosen at or near the lower limits on the decision variables. The procedure is halted if the initial solution does not have a weight-to-buoyancy ratio of less than 1 because any variation in decision variables would necessarily increase the weight of the structure. The initial solution values have to be increased for larger diameter vehicles because a solution in the FEA could not be reached with the specified manufacturing minimums.

The specific code implementation of MADS used is *nMADS*, which handles multiple objectives. *nMADS* splits the optimization process into two steps. First, the solver identifies the *utopia point*, wherein each objective is minimized. This step provides a point on the Pareto front for each objective in the formulation. This single objective solver is called *NOMADm*. Next, points are found to fill the gaps in the Pareto front; in this case 50 points are allocated to fill in the gaps of the Pareto front. For the problems run in this research, however, the Pareto front is rarely generated due to the limited range of valid solutions based on the weight-to-buoyancy ratio (a very tight constraint). *nMADS* is used for both problem types as both have multiple objectives.

Testing reveals the optimization code is very sensitive to initial conditions with large diameter vehicles. This sensitivity stems from the limited search and poll points used in the solver. If additional search and poll points are allocated to MADS, any initial condition converges to the same optimal point. To fix this issue, if a run of MADS produces poor results, the best result found is the initial condition for another run with the previous points kept in memory. This allows the optimizer to find good solutions while avoiding computational effort on previous decision vectors already examined.

Some segments of the nMADS code are shown in Appendix A. In A.7, the number

of objectives, search points, poll points, indifference zone, and initial conditions are specified. The number of objectives and initial conditions changes based on the problem type. A.8 lists the linear constraints of each problem type. A.9 describes the code that calls the A.1 or A.2 file based on problem type. This file also outputs data to the MATLAB window to observe the progress of the code as it runs. A.10 is the MATLAB function that gives the objective functions and nonlinear constraints to the nMADS solver. A.13 shows a section of the Python script that creates data files for the stress and deflection values.

3.3 ABAQUS ® Calls

Finite-element methods can be simplified conceptually using linear algebra. In a structural mechanics case, as in this research, the FEA solver is to compute a vector $\mathbf{u}$ such that $A(\mathbf{u}) \times \mathbf{u} = \mathbf{b}$, where $\mathbf{u}$ is nodal displacement, $A(\mathbf{u})$ is the element stiffness matrix, and $\mathbf{b}$ is the mechanical force vector [43]. Since the analysis of the hexakis icosahedron is nonlinear, due to the displacement of the membrane being larger in magnitude than the thickness of the membrane, the FEA solver attempts to find a vector $\mathbf{u}$ such that $A(\mathbf{u}) \times \mathbf{u} - \mathbf{b} = \mathbf{0}$. This becomes a root finding problem that is sensitive to step size for convergence and run-time properties.

FEA solvers require boundary conditions to solve for displacements and stresses. Boundary condition three (BC3) from previous work in the vacuum LTAVs was used for this FEA [42]. This boundary condition places a restriction on the displacement of a top and bottom node to zero, ensuring symmetry of force and deflection on the object. The symmetrical loading condition is what the object would most likely experience in flight.

ABAQUS ® allows for multiple cores and variable memory to be used in its analysis. With 10 cores at 3 GHz and 24 GB of RAM, problem instances took about

15 minutes to run. A.1 describes the parts of the setup file for the FEA when running Problem Type One. A.2 is the setup file for Problem Type Two. These files show how we implement the material properties, the type of FEA analysis, and how the iteration is recorded in a text file.

The output of the analysis is found using the functions in A.5 and A.6. We consider a payload weight in the analysis and this weight is included in the weight-to-buoyancy ratio as seen in A.6. A.5 demonstrates that if the 100 iterations in ABAQUS ® fail to converge to an answer, the FEA is halted and the output variables are set to create objective values of infinity. This moves the optimizer away from that region of the design space and prevents using computing time to create output files for a null result.

Some factors in the FEA code are kept constant for this analysis. Individual beams are not adjustable, due to manufacturing constraints. We assume a uniform beam radius and thickness for all beams in the hexakis icosahedron. The FEA method used for the research is a Static General step with automatic increments with the addition of nonlinear effects. Nonlinear theory applies to the hexakis because the displacement of the membrane is greater than the thickness of the membrane. This step type is used for both problem types. The increment used was 10^{-2} for all runs.

3.4 High Performance Computing (HPC)

The Air Force Research Laboratory (AFRL) sponsors a Department of Defense (DoD) Supercomputing Resource Center (DSRC). With the hexakis icosahedron problem taking 15 to 20 minutes per iteration, the high performance computing (HPC) offered by the DSRC provides an option to solve these problems. HPC, typically referred to as a *supercomputer*, is able to run many more search and poll steps than a normal desktop computer in a shorter amount of time. The *thunder* platform with one node (36 cores) is used for this research.

The thunder platform interfaces with the optimization code using a different process in comparison to a Windows desktop machine. Only the ABAQUS ® input file is required to run the FEA. A.3 and A.4 are generated by combining previous MATLAB files together from a typical desktop run. A.3, the setup file for the HPC, produces an input file for ABAQUS ® that can be read on the HPC. A.4, the results file, calculates the displacements, stress, and weight-to-buoyancy using the output database returned from the HPC. Parts of the Python scripts used in the creation of the input file and for getting results are shown in A.11 and A.12, respectively. This setup allows the user to run one vehicle instance at a time on the HPC. Running the entire *nMADS* code structure is currently being developed.

3.5 Program Flow and Summary

The coding diagram is shown in Figures 7 and 8. Figure 8 is represented in Figure 7 as the ABAQUS FEA box. For both problem types the process is identical. First, the initial conditions of the design vector, search points, and poll points are provided to MADS. Then, MADS sends the current design vector to the ABAQUS MATLAB files. These files represent the characteristics of the vehicle in an input file, send the input file to the ABAQUS ® solver, and receive the results of the analysis. These results are sent back to MADS, wherein the next design vector is determined. The process continues until one of the stopping criteria is met.

This chapter provided the details of the implementation and integration of the various methodological components used to carry out the design analysis of the hexakis icosahedron LTAV being studied. In Chapter IV, the results of this methodology are shown and compared for both problem types.

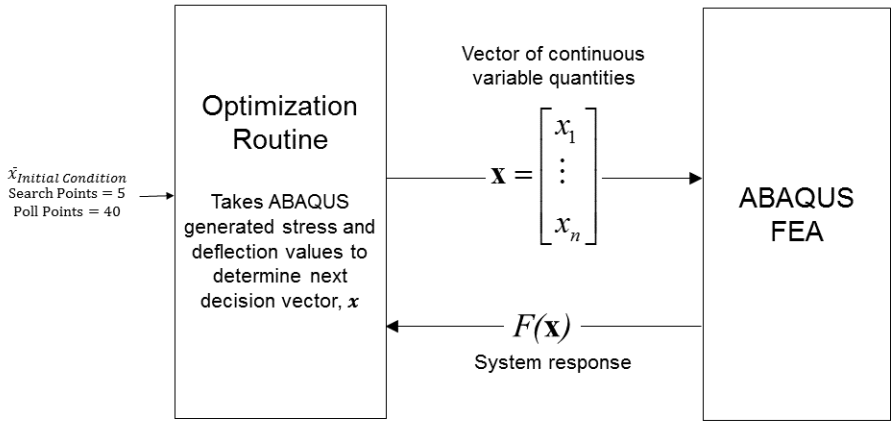


Figure 7. Process Flow

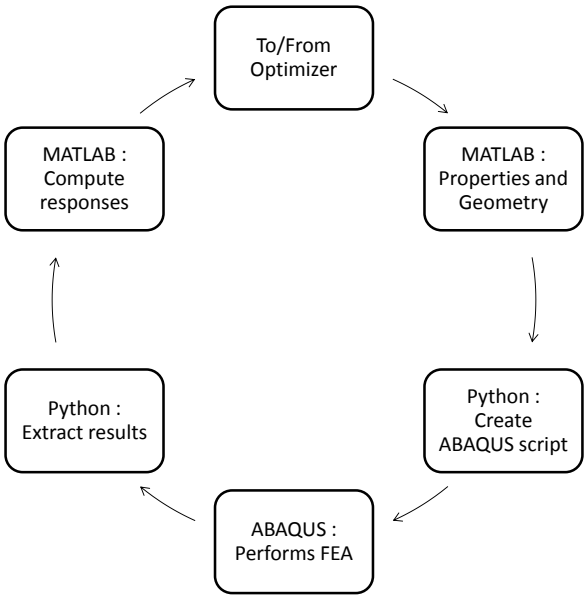


Figure 8. ABAQUS Flow

IV. Results and Discussion

The results of the analysis for each problem type are presented in this chapter. These results follow from the methodology previously described. Problem Type One results are discussed first, followed by Problem Type Two. A new material for the membrane is examined, and the analysis of this updated design is shown at the end of the chapter.

4.1 Problem Type One - Beam and Membrane Optimization

For the first problem type, the decision variables are beam radius, beam thickness, and skin thickness. Carbon nanotubes (CNTs) constitute the frame material with Spectra as the membrane for all-diameter vehicles. Other design particulars are presented as varied.

One-Foot Design.

The initial design under analysis has a one-foot (0.3048 meter) diameter. With a one-foot (0.3048 meter) diameter structure, very little buoyancy exists because the vehicle displaces such a small quantity of air. Therefore, the initial solution to the optimization begins at the manufacturing limits to determine if the one-foot (0.3048 meter) diameter is even feasible. If the manufacturing limits on the beam geometry and skin geometry do not produce a feasible design in terms of weight-to-buoyancy, this size vehicle cannot be made with the CNT frame and Spectra membrane because the structural weight cannot be reduced. Figures 9 and 10 show the stress and deflection results for the one-foot vehicle.

The one-foot (0.3048 meter) diameter hexakis meets the stress limits given the inputs; however, the design has a very high weight-to-buoyancy ratio. The material

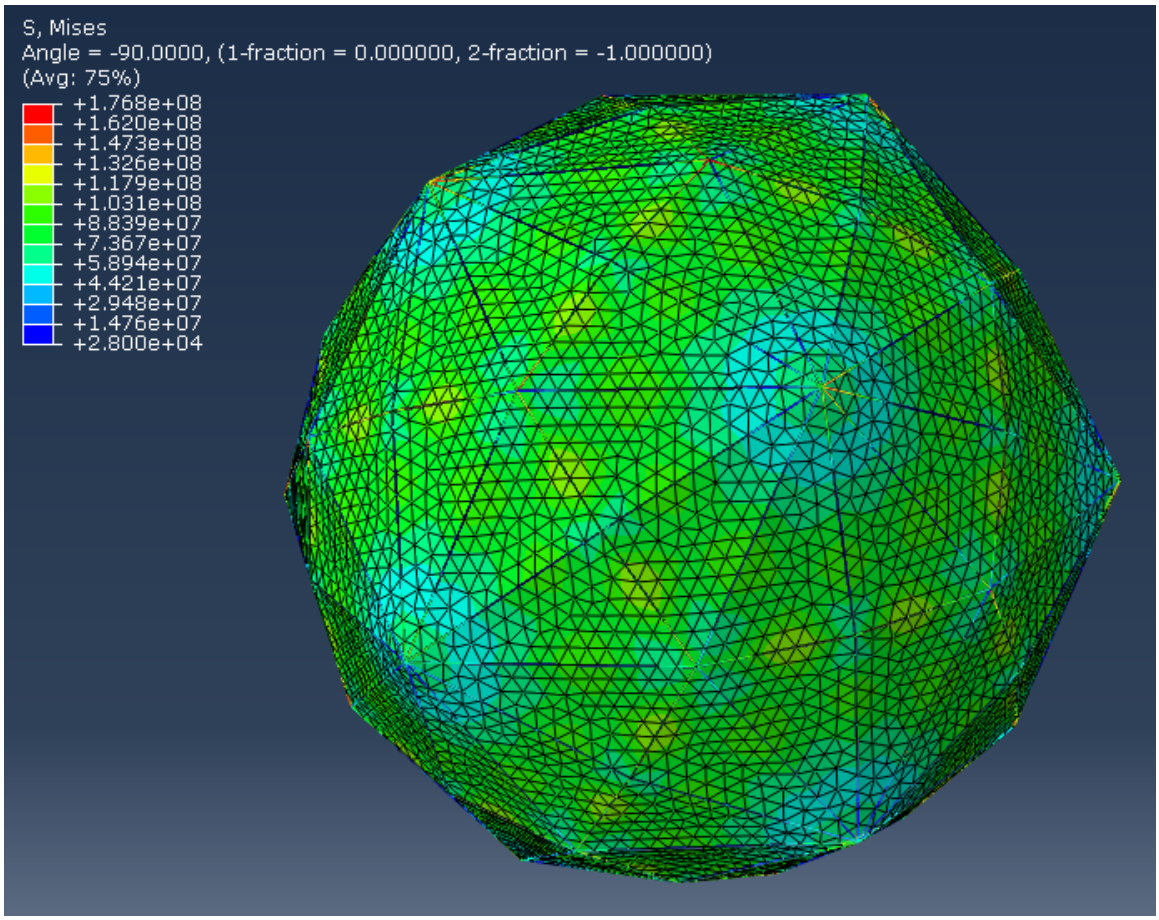


Figure 9. One-Foot (0.3048 Meters) Diameter Stress Results

used to create the structure returns a weight-to-buoyancy ratio of 15.3227. This means that with the materials chosen and manufacturing limits, a one-foot (0.3048 meter) diameter hexakis is not feasible. The buoyancy produced is in the order of ten grams while the structure weighs 217 grams. The disparity between the buoyancy and weight indicates that a much larger vehicle is required to achieve a weight-to-buoyancy ratio less than 1.

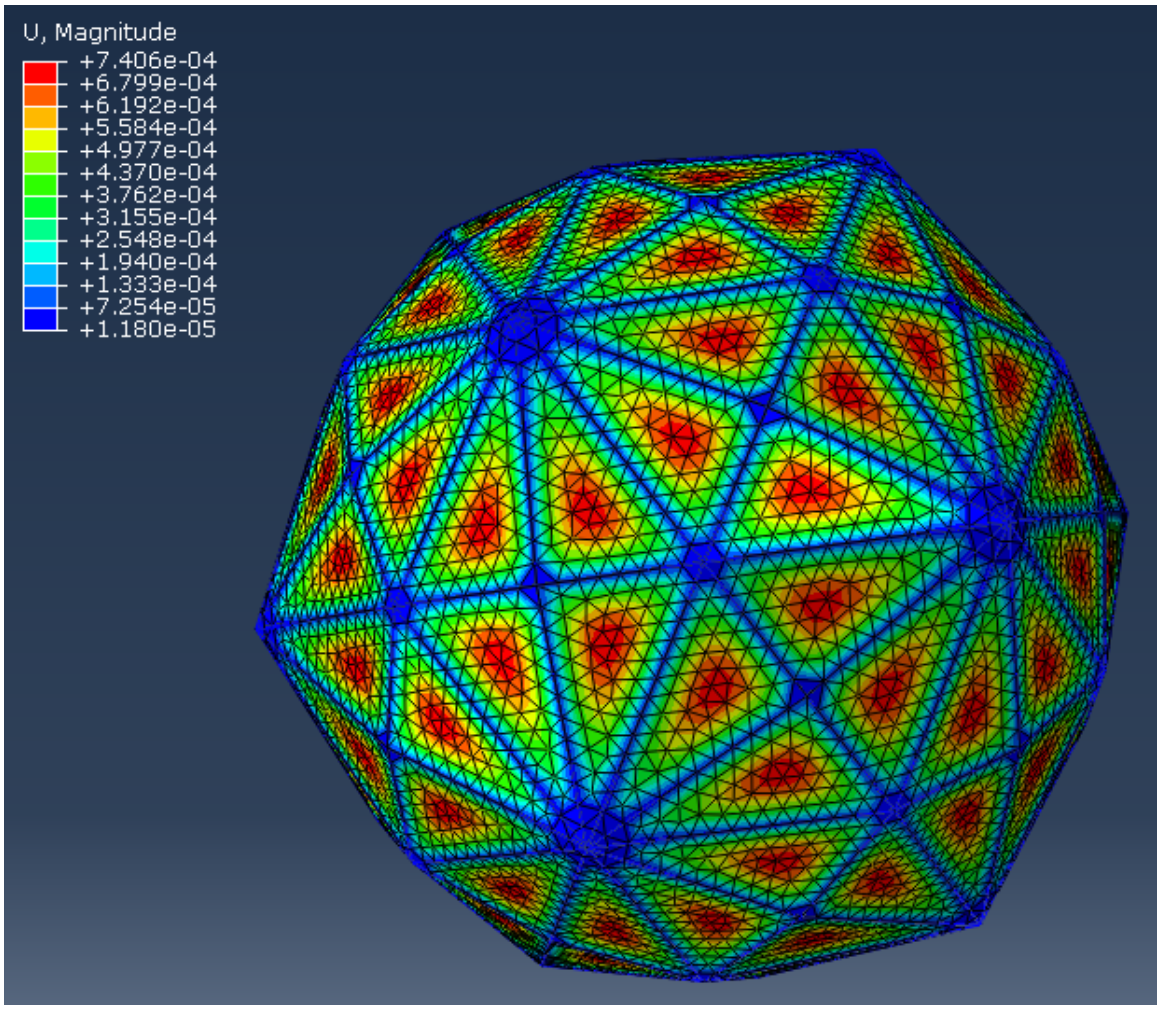


Figure 10. One-Foot (0.3048 Meters) Diameter Displacement Results

Four-Foot Design.

The diameter of the hexakis is now set as four feet (1.2192 meter) to check for positive buoyancy. Again, the structure is tested at manufacturing limits to establish if the design is even feasible. The stress and displacement are shown in Figures 11 and 12, respectively.

The weight-to-buoyancy ratio is 1.6263, still too large to produce a floating structure. The weight of the structure is 1.4 kilograms with 700 grams each for the frame and skin material. The buoyancy is about 890 grams.

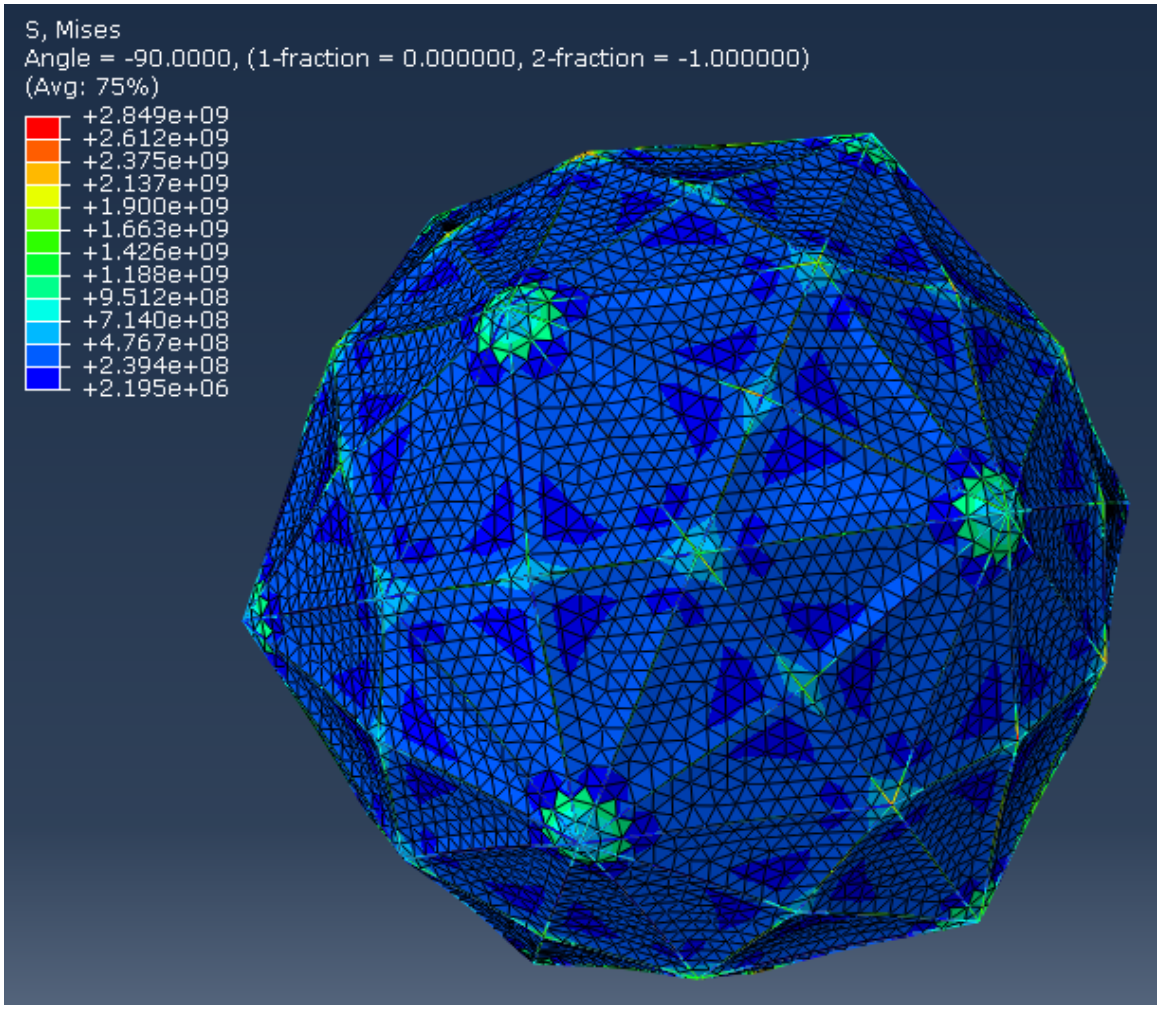


Figure 11. Four Foot (1.2192 Meters) Diameter Stress Results

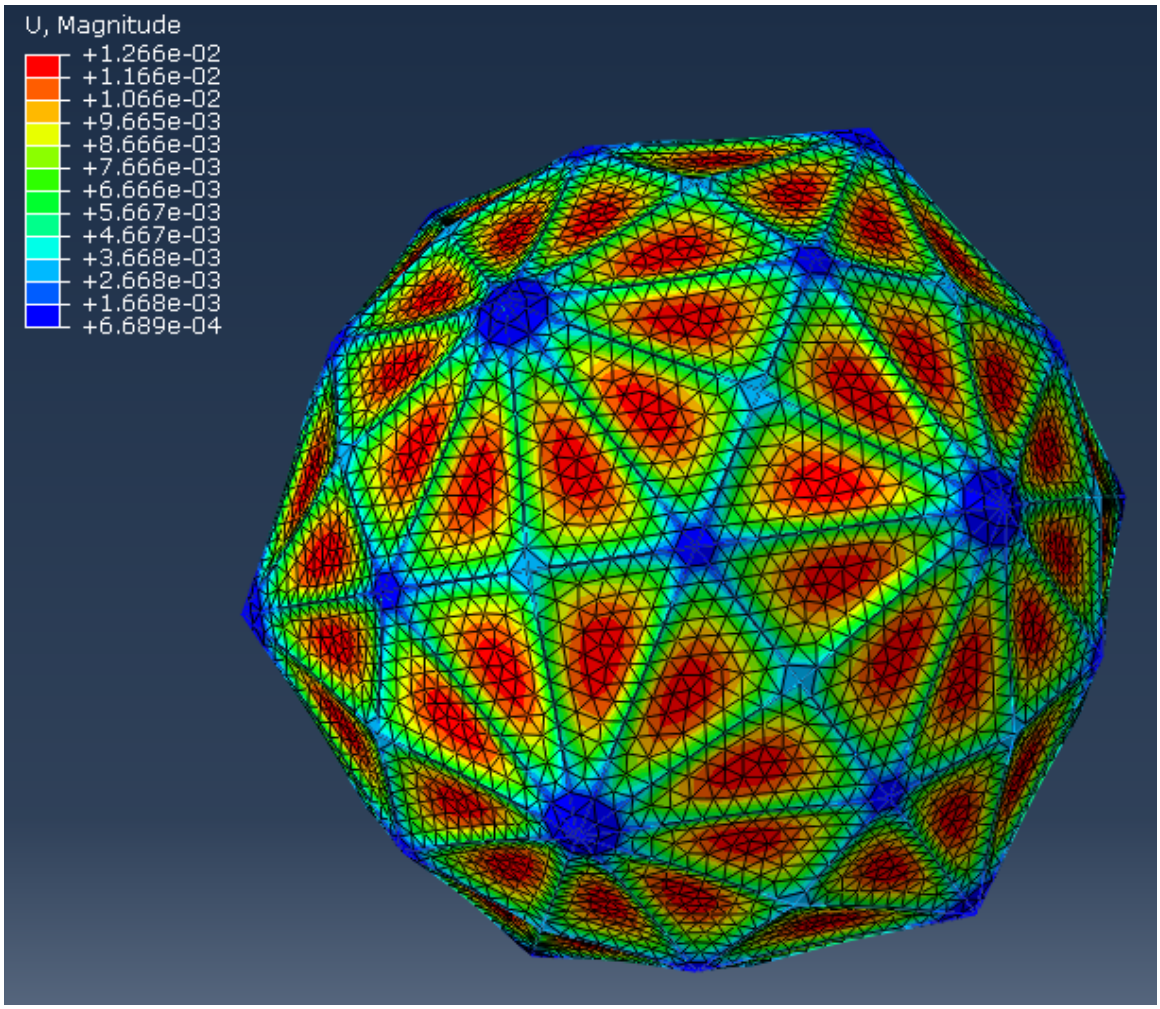


Figure 12. Four Foot (1.2192 Meters) Diameter Displacement Results

Ten-Foot Design.

The decrease in weight-to-buoyancy from the one-foot (0.3048 meter) design to the four-foot (1.2192 meter) design indicates a feasible design is being approached. However, the diameter of the vehicle has to be increased to ten feet (3.048 meters) to handle the extra beam and skin material. At a vehicle radius of ten feet (3.048 meters), the beam radius is increased to produce designs that would solve in ABAQUS ®. Designs at the manufacturing limits do not converge, most likely due to the deflections of the materials being too large. The initial solution is then set as the skin thickness at its manufacturing tolerance, the beam radius at 0.1 meters, and the beam thickness

at 0.002 meters to account for the c-ratio constraint.

Using the optimization code, the best weight-to-buoyancy ratio is found. For the ten-foot design (3.048 meters), the weight-to-buoyancy is 1.4352 with a beam radius of 0.02676 meters, a beam thickness of 0.00054 meters, and a skin thickness of 0.0002 meters. The stress and deflection maximums increase, but are still feasible, as demonstrated in Figures 13 and 14.

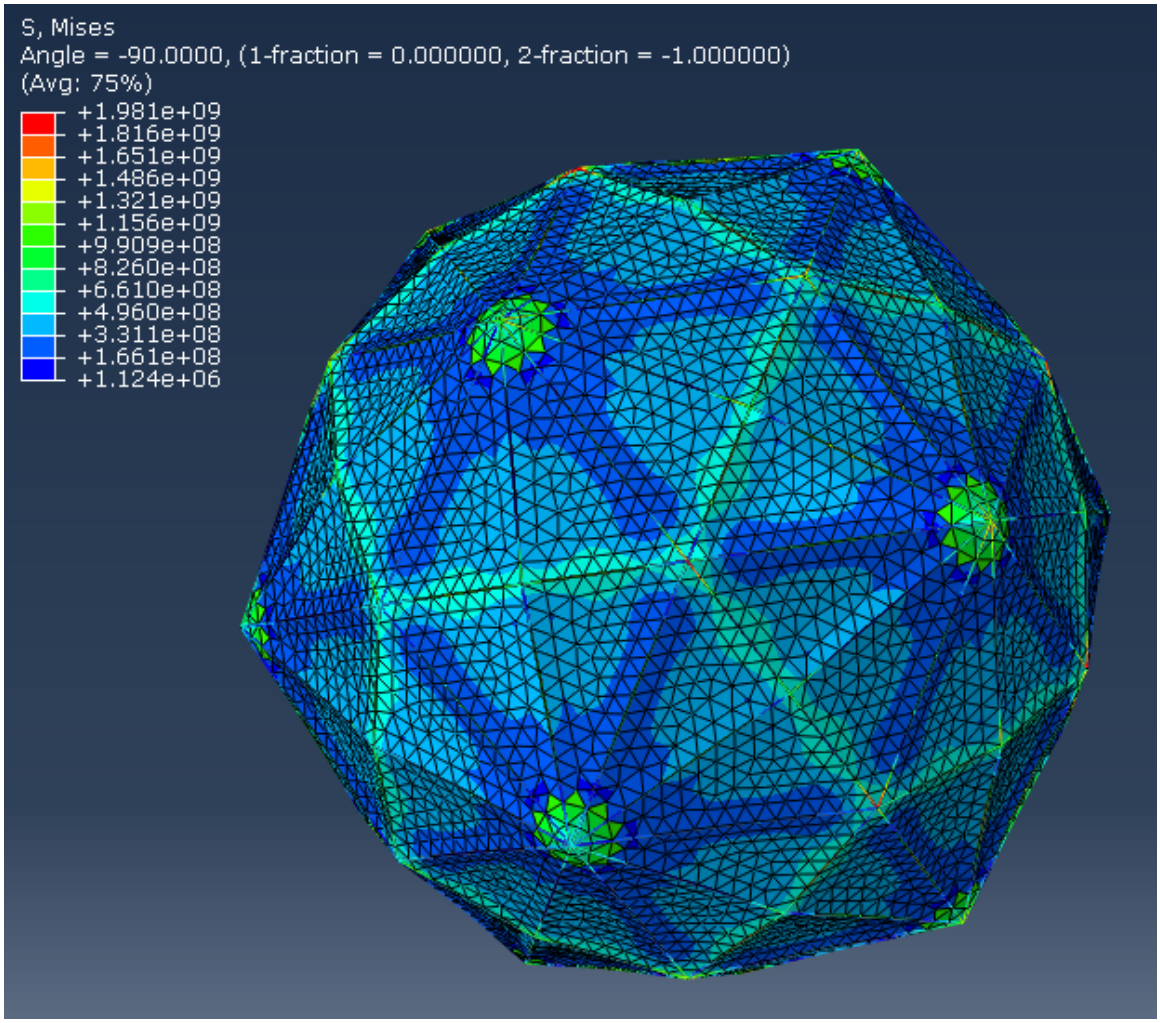


Figure 13. Ten Foot (3.0480 Meters) Diameter Stress Results

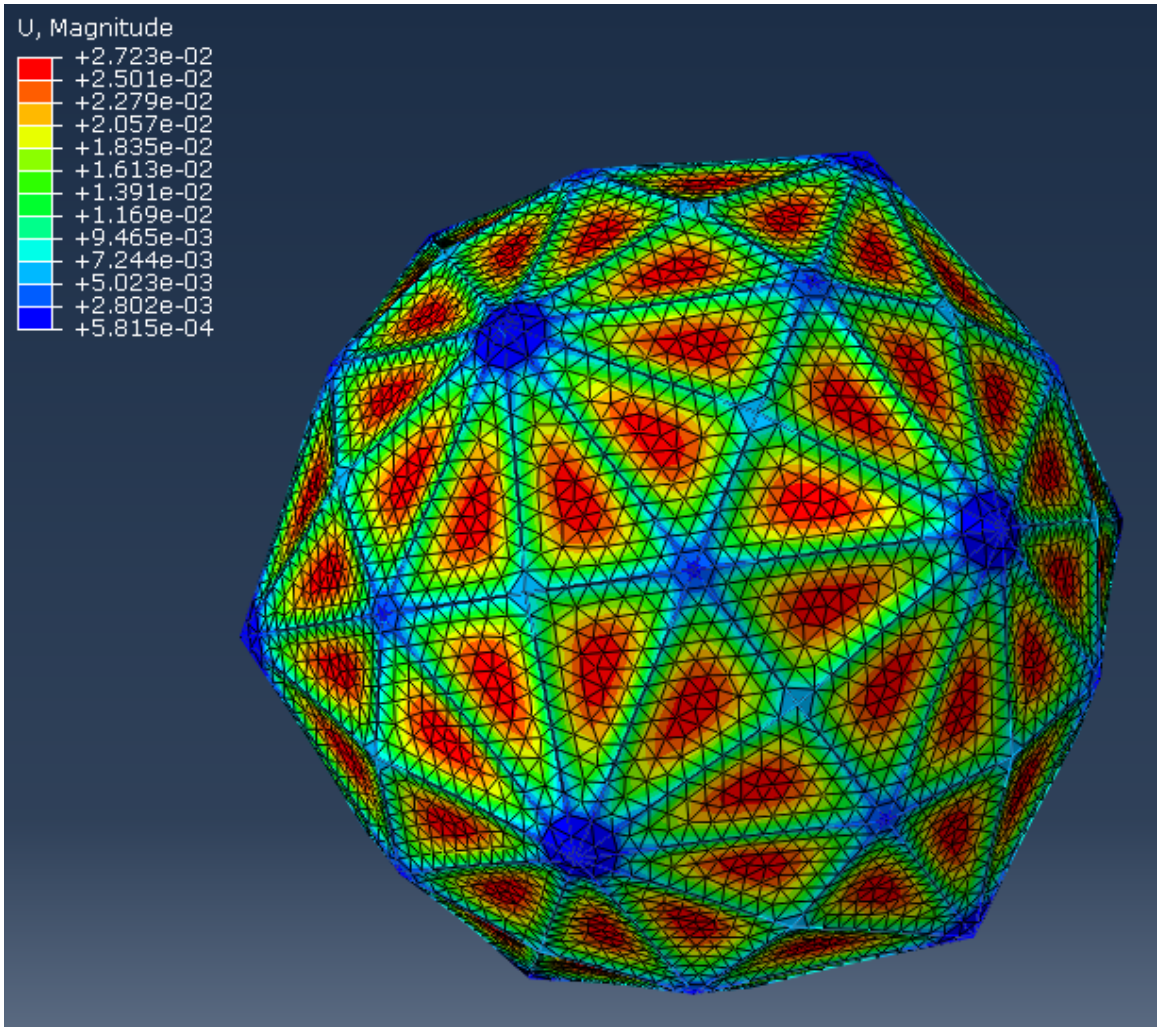


Figure 14. Ten Foot (3.0480 Meters) Diameter Displacement Results

Fifteen-Foot Design.

The next diameter investigated is fifteen feet (4.5720 meter). With this design, it is observed that the problem is very sensitive to the initial solution provided. When the optimization starts with an initial condition identical to the ten foot (3.0480 meter) design, the optimization code converges to a suboptimal answer. This sensitivity may be due to the low number of search and poll points performed in the optimization process or convergence to a local optimal. A small number of points is used because of the time required to perform one iteration (about 20 minutes).

A feasible and good design is found using an initial condition of 0.05 meters as the beam radius, 0.001 meters as the beam thickness, and 0.0002 meters as the skin thickness. A weight-to-buoyancy of 0.9907 is achieved with a final beam radius of 0.0302 meters, beam thickness of 0.000678 meters, and skin thickness of 0.000255 meters. This design meets the required stress and deflection conditions and floats at sea level. The stress and deflection values are shown in Figures 15 and 16.

The frame mass for this design is 32.6 kilograms with a skin weight of 13.9 kilograms and a buoyancy of 46.9 kg. The maximum frame deflection is 26.6 millimeters and the maximum skin deflection is 58.5 millimeters. The maximum frame and skin stresses are 3.3×10^9 Pascals and 1.93×10^9 Pascals respectively. The maximum altitude that can be obtained with this vehicle is 310 feet (95 meters) without a payload. At sea level, the structure can support up to 400 grams.

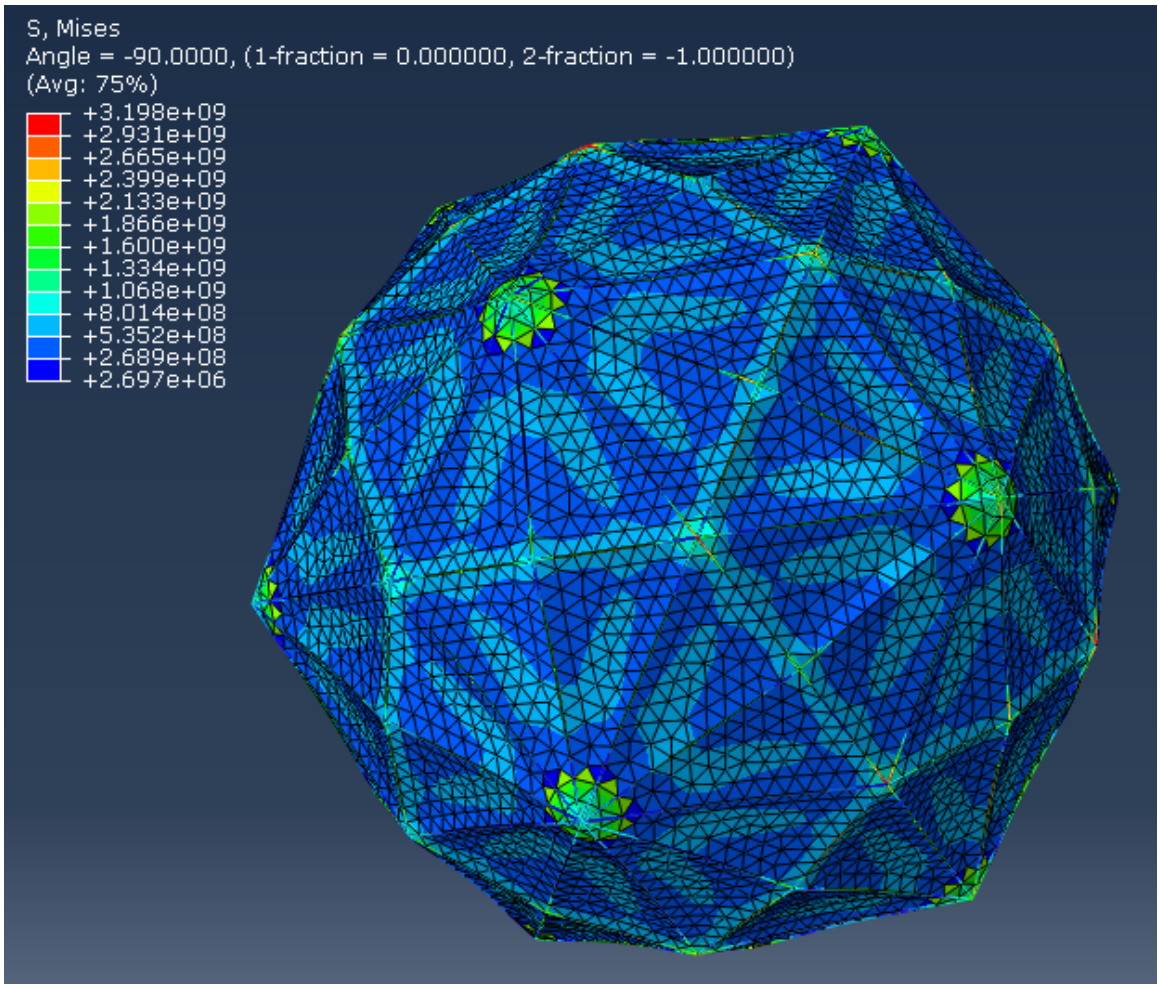


Figure 15. Fifteen Foot (4.5720 Meters) Diameter Stress Results

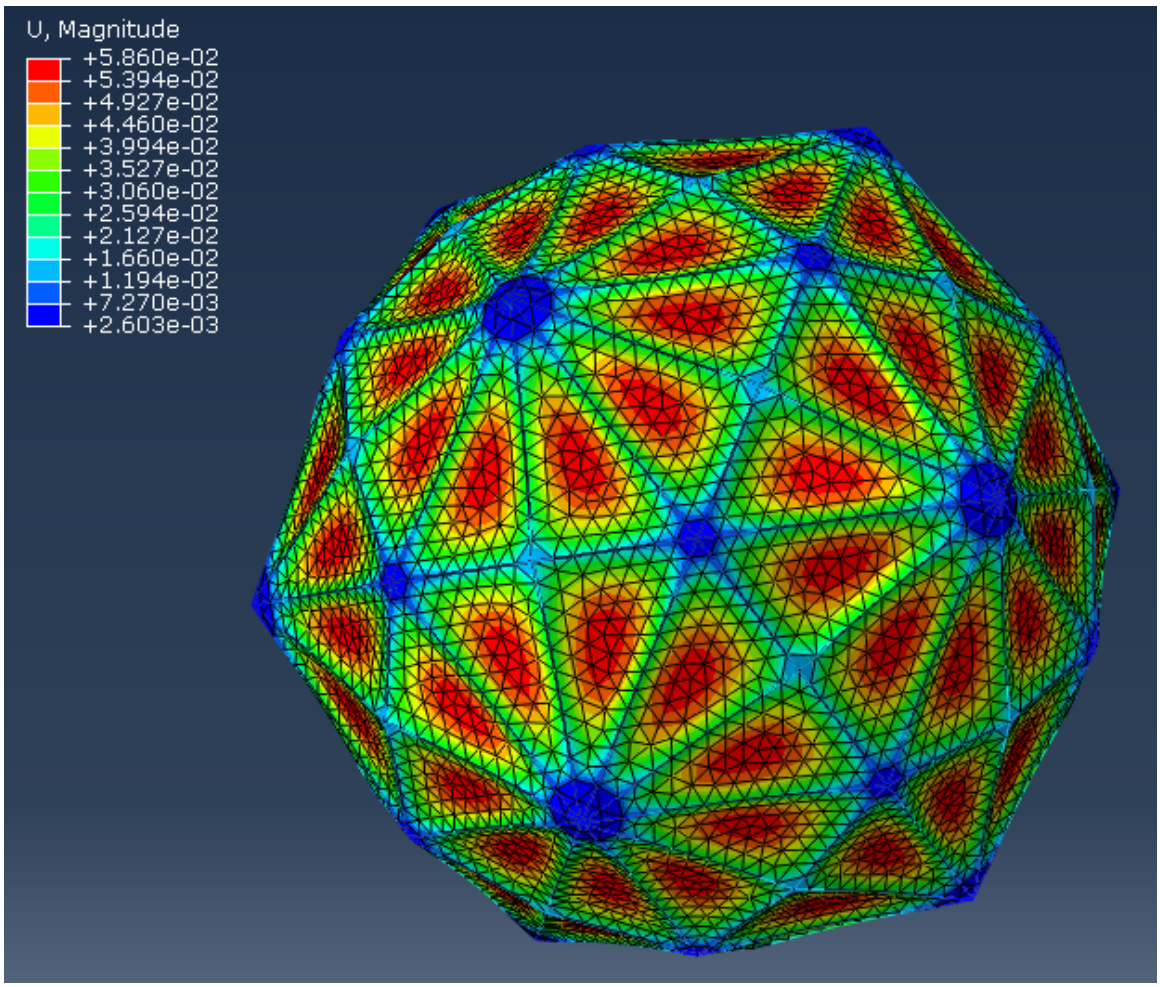


Figure 16. Fifteen Foot (4.5720 Meters) Diameter Displacement Results

Summary of Problem Type One.

The optimization using the Problem Type One formulation confirmed Cranston's findings that a hexakis icosahedron needs to have a large diameter to float when composed of a frame of CNTs and a membrane of Spectra. It is desired to find smaller designs, so an investigation into the material density required to have viable small-diameters is conducted. This leads to the investigation of Problem Type Two.

4.2 Problem Type Two - Material Optimization

This problem formulation changes the material densities to generate a feasible design for a specified radius. The objectives in this problem are to maximize density such that the constraints are met. Vehicle designs of one-foot (0.3048 m), two feet (0.6096 m), five feet (1.5240 m), and ten feet (3.048 m) are examined. The results are shown in Figures 17 through 20. Each figure gives the beam and skin geometry values along with the altitude. As previously discussed, with large vehicle diameters the beams have to be increased in size for the FEA solution to converge. No payload is considered for this analysis.

These figures indicate the relationship between density and weight-to-buoyancy is linear. The linear relationship is appropriate when examining Equation 4 on page 21.

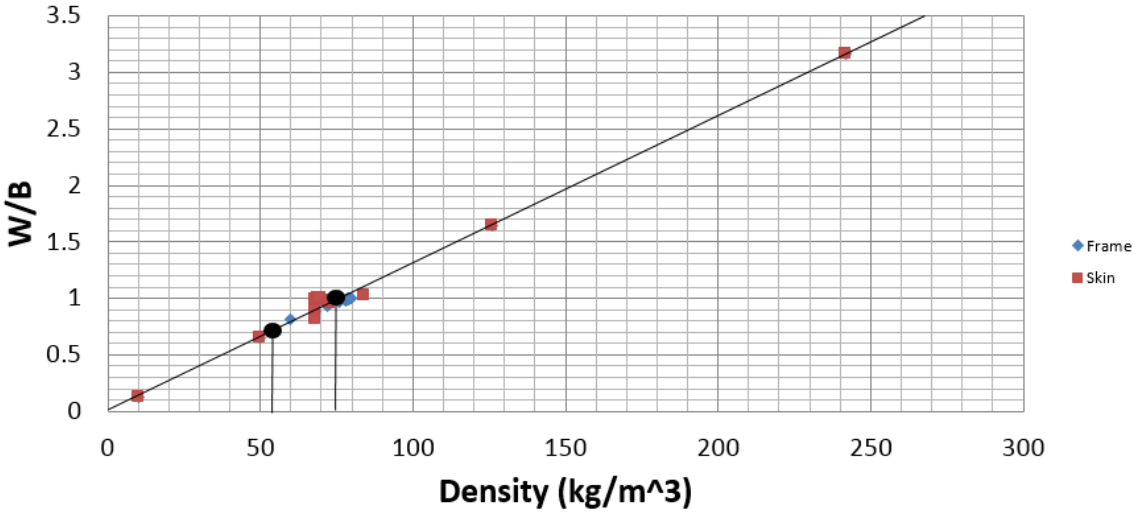


Figure 17. One-Foot (0.3048 Meters) Diameter Material Property Results (Beam Radius = 0.0080 Meters, Beam Thickness = 0.0002 Meters, Skin Thickness = 0.0002 Meters, Altitude = 0 Meters)

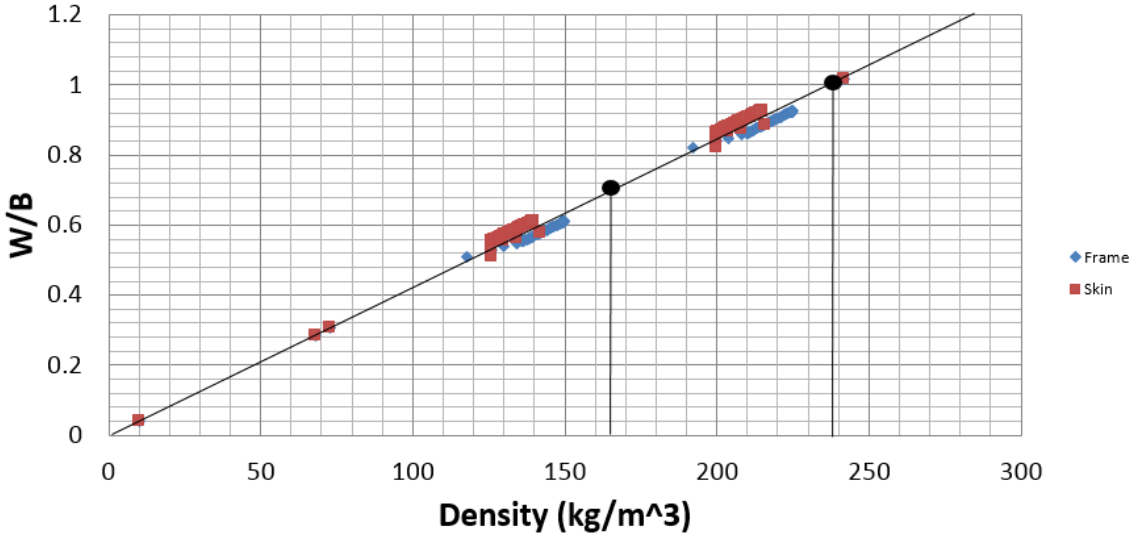


Figure 18. Two Foot (0.6096 Meters) Diameter Material Property Results (Beam Radius = 0.0080 Meters, Beam Thickness = 0.0002 Meters, Skin Thickness = 0.0002 Meters, Altitude = 0 Meters)

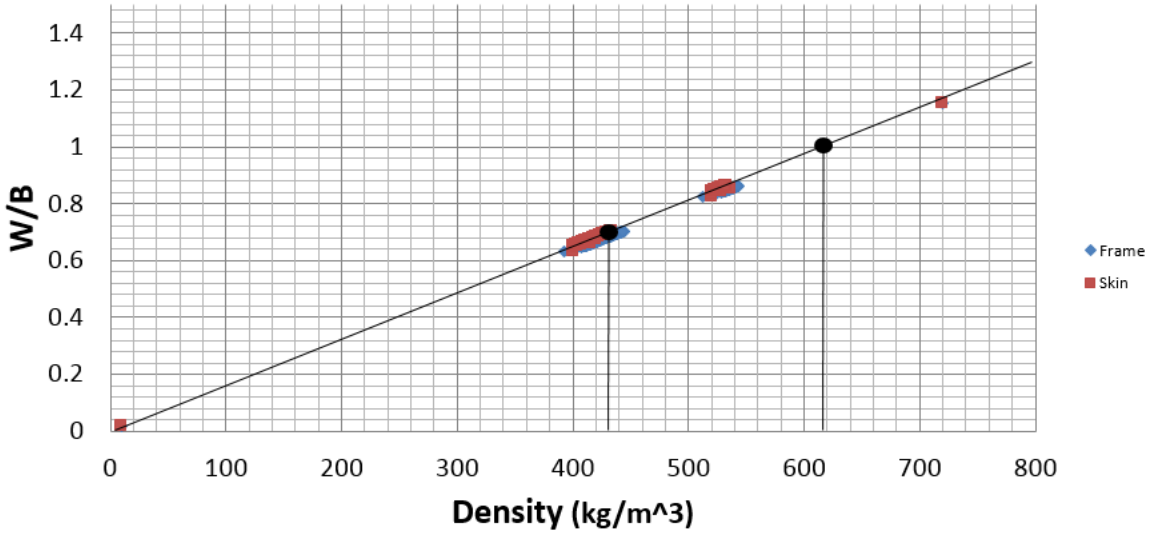


Figure 19. Five Foot (1.5240 Meters) Diameter Material Property Results (Beam Radius = 0.0120 Meters, Beam Thickness = 0.0003 Meters, Skin Thickness = 0.0002 Meters, Altitude = 0 Meters)

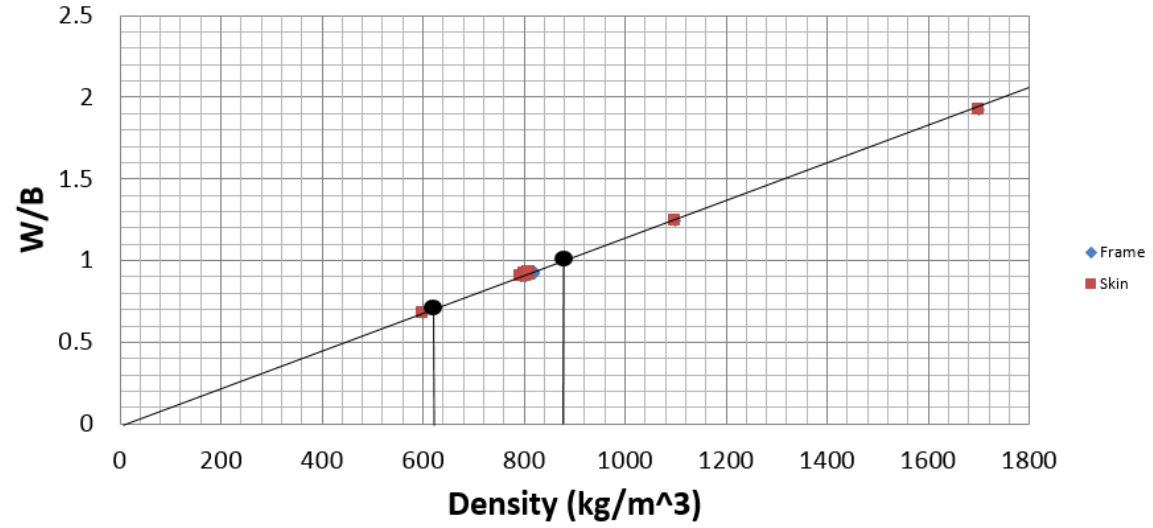


Figure 20. Ten Foot (3.048 Meters) Diameter Material Property Results (Beam Radius = 0.0250 Meters, Beam Thickness = 0.0005 Meters, Skin Thickness = 0.0002 Meters, Altitude = 0 Meters)

Summary of Problem Type Two.

The results of the material optimization are shown more concisely in Table 4. The density required for floating vehicles decreases as vehicle diameter decreases. The column for density required for a weight-to-buoyancy ratio less than 1 is given for designs that can support a small payload. These densities are identical for the frame and skin material. It should be noted that changes in the manufacturing capabilities would alter these results.

Table 4. Material Optimization Results

Diameter (ft, m)	**Beam Radius (mm)**	**Beam Thickness (mm)**	**Skin Thickness (mm)**	**Density for W/B = 1 (kg/m^3)**	**Density for W/B = 0.7 (kg/m^3)**
$1, 0.3048$	8	0.2	0.2	75	55
$2, 0.6096$	8	0.2	0.2	225	165
$5, 1.524$	12	0.3	0.2	620	430
$10, 3.048$	25	0.5	0.2	880	620

The results show that the density of materials is the primary factor in creating LTAVs. With the current manufacturing limitations and material densities, the vehicle sizes here remain infeasible. Therefore, new materials need to be considered for the frame or membrane. After consultation with the Air Force Research Laboratory, graphene is chosen as a contender for the membrane material due to its manufacturing capability and strength.

4.3 Problem Type One - Graphene

With the skin material changing to graphene, the manufacturing limits on the skin thickness and the sizing of the vehicles changes dramatically. Graphene has a minimum skin thickness of 0.33 nanometers, compared to Spectra's 0.2 millimeters. Graphene's properties are shown in Table 5. Graphene is more dense than Spectra and has a lower specific strength, but can be manufactured much thinner. This reduces

the weight of the vehicle to essentially just the frame. The lower specific strength is accounted for by having the membrane include multiple layers of graphene.

Table 5. Graphene Properties

Density (kg/m^3)	**Poisson's Ratio**	**Modulus of Elasticity (GPa)**	**Yield Stress (GPa)**
2000	0.10	500	50

Using graphene as the membrane material, the Problem Type One formulation is used to find an optimal vehicle. The only change in the formulation is the skin thickness constraint. A diameter of four feet (1.2192 meter) is used for this vehicle. The diameter is chosen through experimentation. Smaller diameters were attempted, but the weight-to-buoyancy ratio did not produce a positively buoyant design until the diameter increased to four feet (1.2192 meters). For this design, the beam radius is 8 millimeters with a beam thickness of 0.2 millimeters. The skin thickness is 500 nanometers. The stress and deflection results for the graphene hexakis icosahedron are shown in Figures 21 and 22.

The maximum frame stress is 3.33×10^9 Pascals and the maximum skin stress is 2.25×10^{10} Pascals. Those stresses are 88% and 45% of the material yield stresses respectively. The maximum frame deflection is 7.5 millimeters along with a maximum skin deflection of 26.9 millimeters. The weight of the frame is 0.6771 kilograms and the weight of the graphene membrane is 0.004 kilograms for this vehicle. The design has a weight-to-buoyancy ratio of 0.7654, indicating a payload can be added to the vehicle.

Investigations into the payload capacity and altitude limits are conducted on the graphene hexakis icosahedron. Due to the difference between the weight and buoyancy of the vehicle, the payload has a maximum weight of 200 grams. Without a payload, the structure can float to an altitude of 6900 feet (2100 meters). With a 100 gram payload, the structure has a maximum altitude of 3600 feet (1100 meters).

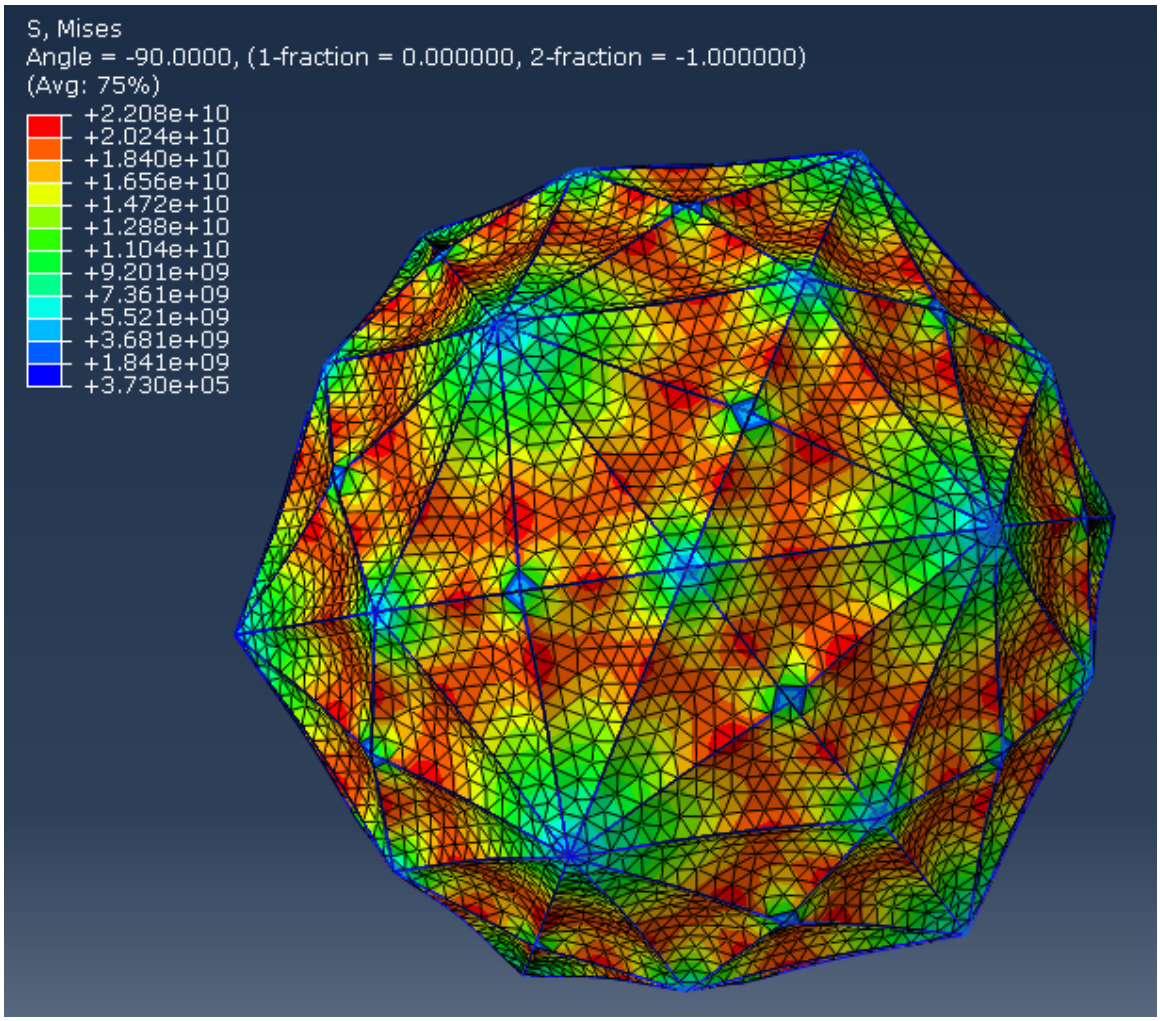


Figure 21. Four Foot (1.2192 Meters) Diameter Graphene Stress Results

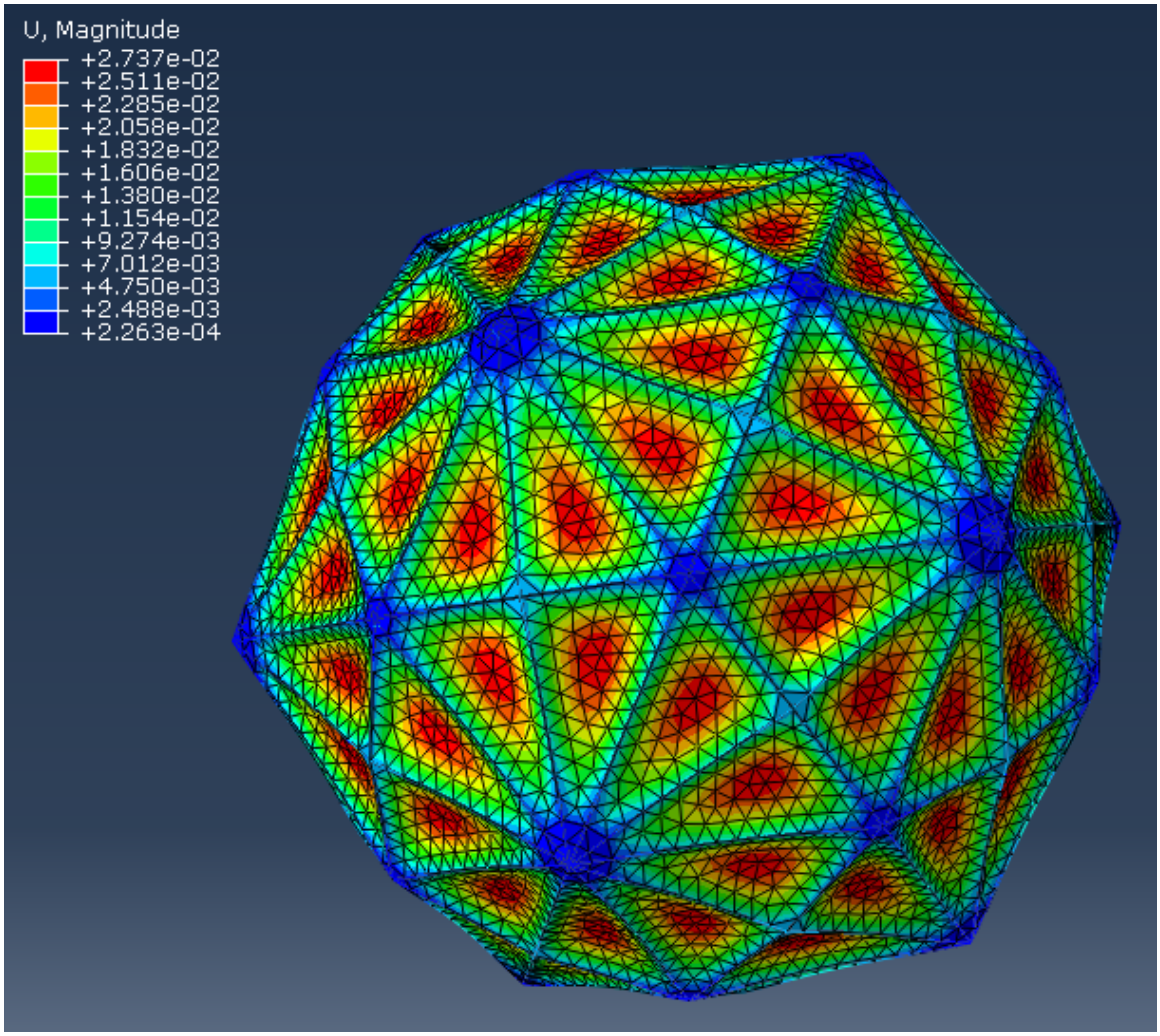


Figure 22. Four Foot (1.2192 Meters) Diameter Graphene Displacement Results

4.4 Design Characteristics and Observations

Some structural characteristics have been discovered through this analysis. The materials are observed to increase stiffness as the pressure increases. However, they never exceeded the yield stress limits on the two feasible designs found. If the skin thickness is made too thin, excess yield or buckling occurs that prevents the FEA solver from converging on a solution. The volume reduction for the vehicles due to the vacuum is about 1%, indicating the amount of buoyancy produced by the vehicle does not change significantly due to the membrane and frame deflection.

Some designs are found that satisfy the weight-to-buoyancy requirements, but fail to meet the yield stress constraints. A design for a hexakis icosahedron at sea level, with a diameter of fifteen feet, using CNTs for the frame and Spectra as the membrane yields a design with a weight-to-buoyancy of 0.9. This design has a beam radius of 0.030 meters, a beam thickness of 0.0006 meters, and a skin thickness of 0.0002 meters. However, the frame stress is 5×10^7 Pascals over the yield stress. The excess stress occurs near the vertex, but not directly on it. This is mostly likely due to the compression force the vertex has from the beam members pushing against each other counteracting the bending moment and tension forces.

V. Conclusions and Future Work

Using the MADS algorithm, the optimal design of a hexakis icosahedron to minimize mass and deflection is found. The results indicate that two possible designs are feasible. The first design is a fifteen foot (4.5720 meter) diameter vehicle with a beam radius of 0.0302 meters, beam thickness of 0.000678 meters, and skin thickness of 0.000255 meters. The frame is composed of CNTs with a membrane made of Spectra. This design has a weight-to-buoyancy of 0.9907. The second design is a four foot (1.2192 meter) diameter vehicle made of CNTs and graphene. The beam radius is 8 millimeters with a beam thickness of 0.2 millimeters. The graphene thickness is 500 nanometers. A weight-to-buoyancy of 0.7654 is achieved in this design. These designs improve on the current diameter of 20 feet (6.096 meters).

The graphene hexakis design allows for a small payload and altitude change, unlike the spectra vehicle. The graphene design can hold a 100 gram payload and float up to 3600 feet (1100 meters) while the spectra vehicle has a maximum altitude of 310 feet (95 meters) with no payload.

5.1 Future Research

Some advancements can be made using the results from this research as a foundation. New materials, differing formulations, and construction can be investigated. Both of these designs use materials not common to manufacturing. The CNT composite demonstrates that these materials are increasing in strength and elasticity and, therefore, the materials required to manufacture this vehicle are nearly feasible [44]. New materials like graphene and aerogels indicate that high strength and low density materials will continue to improve. Some designs may be feasible with the addition of patches of membrane on the vertices. The patches would consist of a thicker skin

compared to the rest of the membrane.

Other materials were investigated for use in the hexakis. An innovative material known as an aerogel is a lightweight metal matrix that retains strength characteristics while reducing density drastically. Two aerogels were considered, one made of silica and one using graphene. Some vehicles were designed with the aerogel materials used as the beam structure, however the aerogels did not have the required strength to support the vacuum-induced loads. A possible exploration would be the use of an aerogel as a complete shell of hexakis icosahedron and placing a graphene membrane over the shell. Instead of placing the force on the small beams, the whole shell of the structure would support the vacuum-induced load. The graphene skin would seal the structure for the vacuum.

The formulation from Problem Type One could be updated with additional objectives and constraints. An objective to maximize the radius of the vehicle in addition to the other objectives could allow the optimization code to find the smallest feasible design. Maximizing the radius of the vehicle while minimizing the weight-to-buoyancy ratio, with an upper bound value of one, would produce the smallest vehicle possible using the chosen frame and membrane materials. Constraints could be added to force the design to float above a given altitude with a payload. Instead of yield stress, ultimate stress could be used as the constraint for the frame and skin.

Additional test and analysis must be conducted on how to build a hexakis LTAV. The current minimum size of the structure is four feet (1.2192 meter) in diameter; however, the individual frame beams are quite small. Additive manufacturing could be considered to manufacture the small beams, but the structure may be too large to consider additive methods to create the entire structure at once. Instead, the beams could be manufactured individually with joining segments made for the 10, 6, and 4 beam intersections. This would standardize construction to two to three beam

lengths and three joining structures. Using joining structures at the intersections of the beams at the vertices could prevent cracks. Another manufacturing method would be to additively manufacture hexagon structures with the beams in place and adhere the hexagons together to form the hexakis icosahedron.

Other considerations on the vehicle are the vacuum generation, membrane permeability, and propulsion methods. The vacuum inside the structure could be created by having a few holes in a beam near an intersection and using a vacuum pump at the intersection with a one-way valve. If the membrane is permeable, a vacuum pump would be required to maintain buoyancy. This would reduce the weight available for a payload. Propulsion of the structure must be considered if the objective of the structure is to loiter over areas. A proposed propulsion method uses solar-powered motors with solar panels built into the membrane. The vehicles may be combined into a small formation, with one vehicle providing the propulsion for the group.

The results of this research are being considered for future analysis by other researchers at AFIT. Dynamic analysis on the two feasible designs found is planned. The optimization effort performed in this thesis shows possible vacuum LTAV sizes previously assumed to be impossible.

The goal diameter of 31 inches (0.7874 meters) is near. If the frame material density can be decreased to 690 kg/m^3 or if the beam thickness can be manufactured to 0.132 millimeters the goal diameter vehicle can be created. The beam thickness value assumes the c-ratio constraints still apply. The corresponding beam radius value for the beam thickness of 0.132 millimeters is 6.6 millimeters.

Appendix A. Code Structure

Sections of the code implemented are shown below. The code is not shown in its entirety due to length, the code displayed here shows how each input for the functions are used and lines changed from the code existing prior to this research.

Listing A.1. Beam/Skin Geometry Run ABAQUS Main File

```
function [output_abaqus] = ABAQUS_Main(rb, tb, ts, payload, incr_num, hex_radius,
    hex_alt)

%% Optimization Rutine
% Last updated: Jan 17, 2017
% Edited by Schwemmer, Joseph
% ********************************************************************
%% Geometry and Material Selection
% Material Selection
%       rho     nu      E           Sy      ; % Units: kg/m^3,-,Pa,Pa
mat5 = [1650   0.2    1000e9       10e9    ]; % Nanocyl NANOCYL? NC7000 Thin Multi-Wall
    Carbon Nanotubes, nu aprox: see 'Paper - Study of Poisson Ratios of Graphene and
    Nanotubes' in references
% mat10= [970    0.33   172e9        3.0e9   ]; % Honeywell Spectra? 1000 Fibercl
mat16= [1250   0.33   293e9        3.8e9   ]; %carbon nanotube composite properties from
    ** paper (CNT composite (NCSU))
%% Materials properties from Michael Snure, AFRL/RYDH
mat17= [2000 0.10 500e9 50e9]; %chemical vapor deposition (CVD) graphene (printed to
    0.33 nm)
% mat18= [22 0.30 1e6 10e6]; %graphene aerogel - not hollow (printed to hundreds of
    nm)
% mat19= [3 0.30 10e6 16e3]; %silica aerogel - not hollow (printed to hundreds of nm)
%% Input
I.index = 1;
index = num2str(I.index);
I.filename = ['icosahedron',index]; % I.filename; % .py filename
[rho,~,temp,press]=stdatmo(hex_alt*.3048); %ft to meters for the input
I.payload = payload;
I.scratch_folder = 'Temp Scratch Files'; % used to create the scratch folder and the
    enviroment .env file

% Job Info (Parallel Processing, memory allocation, use of GPUs)
```

```
I.job.num_cores = 10; % # of cores used in the analysis
I.job.memory_usage = 24*1024; %  amount of allocated memory, MB
I.job.num_GPUs = 0; % number of GPUs (graphics processing units) used, 0 for none

% Static Step Info
I.step.buckle = 0; % ON(1) / OFF(0), ON disables others
I.step.stabilization = 1; % stabilization ON(1) / OFF(0), ON w/membrane section, ON diables Riks
I.step.step_type = 0; % use Riks(1), use General(0); use General(0) w/membrane section
I.step.nonlinear_effects = 'ON'; % ON or OFF, ON w/membrane section
I.step.increment_method = 'AUTOMATIC'; % Increments (arc length if Riks) method: 'FIXED' or 'AUTOMATIC'
I.step.maxnuminc = 100; % max number of increments, if fixed

% Static General
I.step.initial_inc = 1e-2; % starting time increment
I.step.max_inc = 1; % max time increment
I.step.min_inc = 1e-36; % min time increment
I.step.stabilization_ratio = 0.05; % w/membrane only - adaptive stabilization: max stabilization/strain energy ratio, default = 0.05
I.step.stabilization_magn = 0.0002; % w/membrane only - dissipated energy fraction, default = 0.0002

% Load and skin sections defined

% Mesh
I.mesh.skin_element_type1 = 'M3D3';%'M3D3' or 'S3'; % See 'Shell and Membrane Element Library Info.txt'
I.mesh.skin_element_type2 = 'M3D3';%'M3D3';
I.mesh.skin_element_shape = 'TRI'; % Element shape: rectangular or triangular
I.mesh.skin_seed_number   =  0.0065*(hex_radius/6);%0.005  ; % skin # of elements/edge, 30 edges in total
I.mesh.frame_element_type = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.frame_seed_number  =  0.0065*(hex_radius/6);%0.005  ; % frame # of elements/edge, 30 edges in total
I.mesh.rays_element_type  = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.rays_seed_number   =  0.0065*(hex_radius/6);%0.005  ; % rays # of elements/edge, 20 edges in total
I.mesh.stiff_element_type = 'B32'; % need to use beam element type: B31, B32, etc.
```

```
I.mesh.stiff_seed_number  =  0.0065*(hex_radius/6);%0.005  ; % rays # of elements/
    edge, 60 edges in total

% Parameters for W/B ratio calculation
I.W_B.rho   = rho; % air density at SL, kg/m^3, http://en.wikipedia.org/wiki/
    Density_of_air
I.W_B.g     = 9.81; % acceleration of gravity, m/s^2
I.W_B.To    = temp; % K, external temp (altitude dependent)
I.W_B.Ti    = I.W_B.To; % K, internal temp (altitude and heat transfer dependent)
I.W_B.Po    = press; % Pa, external pressure (altitude dependent)

%%
% Material Assignment
matf = mat16; % assigned frame material (from the selection above)
mats = mat17; % assigned skin  material (from the selection above)
matr = mat5; % assigned rays  material (from the selection above)
matst= mat5; % assigned stiffners  material (from the selection above)

% Geometry (icosahedron)
I.geometry.structure = 1; % 0 for icosahedron, 1 for hexakis icosahedron, 2 for
    celestial
I.section.hollow_profile_rays = 1; % Rays beam profile: hollow(1),solid(0); beam
    thickness ignored if (0)
I.section.hollow_profile_stiff= 1; % Stiff beam profile: hollow(1),solid(0); beam
    thickness ignored if (0)
I.section.hollow_profile      = 1; % Frame beam profile: hollow(1),solid(0); beam
    thickness ignored if (0)

% Assume hexakis icosahedron, hollow everything
se = (sqrt(15*(85-31*sqrt(5)))/11)*(I.geometry.hexirn/I.geometry.hexir);
me = (3*sqrt(15*(65+19*sqrt(5)))/55)*(I.geometry.hexirn/I.geometry.hexir);
le = (2*sqrt(15*(5-sqrt(5)))/5)*(I.geometry.hexirn/I.geometry.hexir);
s = .5*(se+me+le);
ta = sqrt(s*(s-se)*(s-me)*(s-le));
hexV = (180*(5+4*sqrt(5))/11)*(I.geometry.hexirn/I.geometry.hexir)^3;
I.geometry.initial_volume = hexV;
I.geometry.skin_thickness = ts; % meters
I.geometry.skin_volume = 120*ta*I.geometry.skin_thickness;

I.geometry.frame_beam_radius    = rb; % meters
```

```
I.geometry.frame_beam_thickness = tb; % meters

I.geometry.frame_volume = (pi*60*(2*I.geometry.frame_beam_thickness*I.geometry.
    frame_beam_radius...
    -I.geometry.frame_beam_thickness^2))*(le+me+se);

I.geometry.rays_beam_radius    = 0; % meters
I.geometry.rays_beam_thickness = 0; % meters

I.geometry.stiff_beam_radius    = 0; % meters
I.geometry.stiff_beam_thickness = 0; % meters

% Prints set W/B
str1 = 'icosahedron_properties_';
str2 = int2str(incr_num);
str3 = '.txt';
strT = strcat(str1,str2,str3);
f = fopen(strT,'w');

% Prints Icosahedron Properties
fprintf(f,'\r\nIcosahedron\r\n');
fprintf(f,'-----------------------------------------------------------------------\r\n\r\n
    ');
fprintf(f,'Geometry:\r\n******************************************************\r\n');
fprintf(f,'Icosahedron Radius  : %.4f (m)\r\n',I.geometry.r);
fprintf(f,'Skin Thickness      : %.4e (m)\r\n',I.geometry.skin_thickness);
fprintf(f,'Beam radius         : %.4e (m)\r\n',I.geometry.frame_beam_radius);
fprintf(f,'Beam thickness      : %.4e (m)\r\n',I.geometry.frame_beam_thickness);
fprintf(f,'Rays radius         : %.4e (m)\r\n',I.geometry.rays_beam_radius);
fprintf(f,'Rays thickness      : %.4e (m)\r\n',I.geometry.rays_beam_thickness);
fprintf(f,'Stiffners radius    : %.4e (m)\r\n',I.geometry.stiff_beam_radius);
fprintf(f,'Stiffners thickness : %.4e (m)\r\n',I.geometry.stiff_beam_thickness);
fprintf(f,'Other:\r\n******************************************************\r\n');
fprintf(f,'Payload             : %.4e (kg)\r\n\r\n',I.payload);
fprintf(f,'Altitude            : %.4e (ft)\r\n\r\n',hex_alt);

% Prints Materials Properties
fprintf(f,'\r\nFrame Material Properties:\r\n
    **************************************************\r\n');
fprintf(f,'Density           : %.1f (kg/m^3)\r\n\r\n',matf(1));
```

```
fprintf(f,'Poisson ratio      : %.2f (-)\r\n' ,matf(2));
fprintf(f,'Modulus            : %.4e (Pa)\r\n',matf(3));
fprintf(f,'Yield              : %.4e (Pa)\r\n',matf(4));
fprintf(f,'\r\nSkin Material Properties:\r\n
    **************************************************\r\n');
fprintf(f,'Density            : %.1f (kg/m^3)\r\n',mats(1));
fprintf(f,'Poisson ratio      : %.2f (-)\r\n' ,mats(2));
fprintf(f,'Modulus            : %.4e (Pa)\r\n',mats(3));
fprintf(f,'Yield              : %.4e (Pa)\r\n',mats(4));
fclose(f);

end
```

Listing A.2. Material Property Run ABAQUS Main File

```
function [output_abaqus] = ABAQUS_Main_Mat(frame_rho, skin_rho, incr_num, hex_radius,
     hex_alt)

%% Optimization Rutine
% Last updated: Jan 17, 2016
% Edited by Schwemmer, Joseph
% ********************************************************************
%% Geometry and Material Selection
% Material Selection
%       rho     nu      E           Sy      ; % Units: kg/m^3,-,Pa,Pa
mat5 = [1650   0.2    1000e9       10e9    ]; % Nanocyl NC7000 Thin Multi-Wall Carbon
    Nanotubes
mat13= [skin_rho 0.33 100e9 3.0e9]; % skin material
mat16= [frame_rho 0.33 100e9 3.0e9]; % frame material

%% Input
I.index = 1;
index = num2str(I.index);
I.filename = ['icosahedron',index]; % I.filename; % .py filename
[rho,~,temp,press]=stdatmo(hex_alt*.3048); %ft to meters for the input
I.scratch_folder = 'Temp Scratch Files'; % used to create the scratch folder and the
    enviroment .env file

% Job Info (Parallel Processing, memory allocation, use of GPUs)
I.job.num_cores = 6; %# of cores used in the analysis
```

```
I.job.memory_usage = 16*1024; %  amount of allocated memory, MB
I.job.num_GPUs = 0; % number of GPUs (graphics processing units) used, 0 for none

% Static Step Info
I.step.buckle = 0; % ON(1) / OFF(0), ON disables others
I.step.stabilization = 1; % stabilization ON(1) / OFF(0), ON w/membrane section, ON diables Riks
I.step.step_type = 0; % use Riks(1), use General(0); use General(0) w/membrane section
I.step.nonlinear_effects = 'ON'; % ON or OFF, ON w/membrane section
I.step.increment_method = 'AUTOMATIC'; % Increments (arc length if Riks) method: 'FIXED' or 'AUTOMATIC'
I.step.maxnuminc = 100; % max number of increments, if fixed

% Static General
I.step.initial_inc = 1e-2; % starting time increment
I.step.max_inc = 1; % max time increment
I.step.min_inc = 1e-36; % min time increment
I.step.stabilization_ratio = 0.05; % w/membrane only - adaptive stabilization: max stabilization/strain energy ratio, default = 0.05
I.step.stabilization_magn = 0.0002; % w/membrane only - dissipated energy fraction, default = 0.0002

% Loads and skin sections defined

% Mesh
I.mesh.skin_element_type1 = 'M3D3';%'M3D3' or 'S3'; % See 'Shell and Membrane Element Library Info.txt'
I.mesh.skin_element_type2 = 'M3D3';%'M3D3';
I.mesh.skin_element_shape = 'TRI'; % Element shape: rectangular or triangular
I.mesh.skin_seed_number   =  0.0065*(hex_radius/6);%0.005  ; % skin # of elements/edge, 30 edges in total
I.mesh.frame_element_type = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.frame_seed_number  =  0.0065*(hex_radius/6);%0.005  ; % frame # of elements/edge, 30 edges in total
I.mesh.rays_element_type  = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.rays_seed_number   =  0.0065*(hex_radius/6);%0.005  ; % rays # of elements/edge, 20 edges in total
I.mesh.stiff_element_type = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.stiff_seed_number  =  0.0065*(hex_radius/6);%0.005  ; % rays # of elements/
```

```
    edge, 60 edges in total
54
55 % Parameters for W/B ratio calculation
56 I.W_B.rho   = rho; % air density at SL, kg/m^3, http://en.wikipedia.org/wiki/
       Density_of_air
57 I.W_B.g     = 9.81; % acceleration of gravity, m/s^2
58 I.W_B.To    = temp; % K, external temp (altitude dependent)
59 I.W_B.Ti    = I.W_B.To; % K, internal temp (altitude and heat transfer dependent)
60 I.W_B.Po    = press; % Pa, external pressure (altitude dependent)
61
62 %%
63 % Material Assignment
64 matf = mat16; % assigned frame material (from the selection above)
65 mats = mat13; % assigned skin  material (from the selection above)
66 matr = mat5; % assigned rays  material (from the selection above)
67 matst= mat5; % assigned stiffners  material (from the selection above)
68
69 % Geometry assignments
70
71 % Assume hexakis icosahedron, hollow everything
72 se = (sqrt(15*(85-31*sqrt(5)))/11)*(I.geometry.hexirn/I.geometry.hexir);
73 me = (3*sqrt(15*(65+19*sqrt(5)))/55)*(I.geometry.hexirn/I.geometry.hexir);
74 le = (2*sqrt(15*(5-sqrt(5)))/5)*(I.geometry.hexirn/I.geometry.hexir);
75 s = .5*(se+me+le);
76 ta = sqrt(s*(s-se)*(s-me)*(s-le));
77 hexV = (180*(5+4*sqrt(5))/11)*(I.geometry.hexirn/I.geometry.hexir)^3;
78 I.geometry.initial_volume = hexV;
79 I.geometry.skin_thickness = 0.0002;
80 I.geometry.skin_volume = 120*ta*I.geometry.skin_thickness;
81
82 I.geometry.frame_beam_radius    = 0.025; % meters
83 I.geometry.frame_beam_thickness = 0.0005; % input
84
85 I.geometry.frame_volume = (pi*60*(2*I.geometry.frame_beam_thickness*I.geometry.
       frame_beam_radius...
86     -I.geometry.frame_beam_thickness^2))*(le+me+se);
87
88 I.geometry.rays_beam_radius    = 0; % meters
89 I.geometry.rays_beam_thickness = 0; % input
90
```

```
I.geometry.stiff_beam_radius     = 0; % meters
I.geometry.stiff_beam_thickness = 0; % input

% Prints set W/B
str1 = 'icosahedron_properties_';
str2 = int2str(incr_num);
str3 = '.txt';
strT = strcat(str1,str2,str3);

% Prints material properties to file

end
```

Listing A.3. HPC Setup File

```
function [] = HPC_Create(rb, tb, ts, hex_radius, hex_alt)

% Creates .inp files for use on the HPC
%% Geometry and Material Selection
% Material Selection
%       rho     nu      E            Sy     ; % Units: kg/m^3,-,Pa,Pa
mat5 = [1650   0.2    1000e9       10e9    ]; % Nanocyl NANOCYL? NC7000 Thin Multi-Wall
    Carbon Nanotubes, nu aprox: see 'Paper - Study of Poisson Ratios of Graphene and
    Nanotubes' in references
mat13= [970    0.33   172e9        3.0e9   ]; % enhanced membrane properties (spectra
    1000 fiber)
mat16= [1250   0.33   293e9        3.8e9   ]; %carbon nanotube composite properties from
    ** paper (CNT composite (NCSU))
%% Input

I.index = hex_radius*2;
index = num2str(I.index);
I.filename = ['icosahedron',index]; % I.filename; % .py filename
[rho,~,temp,press]=stdatmo(hex_alt*.3048); %ft to meters for the input
I.scratch_folder = 'Temp Scratch Files'; % used to create the scratch folder and the
    enviroment .env file

% Static Step Info
I.step.buckle = 0; % ON(1) / OFF(0), ON disables others
I.step.stabilization = 1; % stabilization ON(1) / OFF(0), ON w/membrane section, ON
```

```
    diables Riks
I.step.step_type = 0; % use Riks(1), use General(0); use General(0) w/membrane
    section
I.step.nonlinear_effects = 'ON'; % ON or OFF, ON w/membrane section
I.step.increment_method = 'AUTOMATIC'; % Increments (arc length if Riks) method: '
    FIXED' or 'AUTOMATIC'
I.step.maxnuminc = 1000; % max number of increments, if fixed
% Static General
I.step.initial_inc = 1e-3; % starting time increment
I.step.max_inc = 1; % max time increment
I.step.min_inc = 1e-36; % min time increment
I.step.stabilization_ratio = 0.05; % w/membrane only - adaptive stabilization: max
    stabilization/strain energy ratio, default = 0.05
I.step.stabilization_magn = 0.0002; % w/membrane only - dissipated energy fraction,
    default = 0.0002

% Load and skin section assignments

% Mesh
I.mesh.skin_element_type1 = 'M3D3';%'M3D3' or 'S3'; % See 'Shell and Membrane Element
     Library Info.txt'
I.mesh.skin_element_type2 = 'M3D3';%'M3D3';
I.mesh.skin_element_shape = 'TRI'; % Element shape: rectangular or triangular
I.mesh.skin_seed_number   =  0.0065*(hex_radius/6);%0.005  ; % skin # of elements/
    edge, 30 edges in total
I.mesh.frame_element_type = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.frame_seed_number  =  0.0065*(hex_radius/6);%0.005  ; % frame # of elements/
    edge, 30 edges in total
I.mesh.rays_element_type  = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.rays_seed_number   =  0.0065*(hex_radius/6);%0.005  ; % rays # of elements/
    edge, 20 edges in total
I.mesh.stiff_element_type = 'B32'; % need to use beam element type: B31, B32, etc.
I.mesh.stiff_seed_number  =  0.0065*(hex_radius/6);%0.005  ; % rays # of elements/
    edge, 60 edges in total

% Parameters for W/B ratio calculation
I.W_B.rho   = rho; % air density at SL, kg/m^3, http://en.wikipedia.org/wiki/
    Density_of_air
I.W_B.g     = 9.81; % acceleration of gravity, m/s^2
I.W_B.To    = temp; % K, external temp (altitude dependent)
```

```
I.W_B.Ti     = I.W_B.To; % K, internal temp (altitude and heat transfer dependent)
I.W_B.Po     = press; % Pa, external pressure (altitude dependent)

%%
% Material Assignment
matf = mat16; % assigned frame material (from the selection above)
mats = mat13; % assigned skin  material (from the selection above)
matr = mat5; % assigned rays  material (from the selection above)
matst= mat5; % assigned stiffners  material (from the selection above)

% Geometry calculations

% Assume hexakis icosahedron, hollow everything
se = (sqrt(15*(85-31*sqrt(5)))/11)*(I.geometry.hexirn/I.geometry.hexir);
me = (3*sqrt(15*(65+19*sqrt(5)))/55)*(I.geometry.hexirn/I.geometry.hexir);
le = (2*sqrt(15*(5-sqrt(5)))/5)*(I.geometry.hexirn/I.geometry.hexir);
s = .5*(se+me+le);
ta = sqrt(s*(s-se)*(s-me)*(s-le));
hexV = (180*(5+4*sqrt(5))/11)*(I.geometry.hexirn/I.geometry.hexir)^3;
I.geometry.initial_volume = hexV;
I.geometry.skin_thickness = ts; % meters
I.geometry.skin_volume = 120*ta*I.geometry.skin_thickness;

I.geometry.frame_beam_radius     = rb; % meters
I.geometry.frame_beam_thickness = tb; % meters

I.geometry.frame_volume = (pi*60*(2*I.geometry.frame_beam_thickness*I.geometry.
    frame_beam_radius...
    -I.geometry.frame_beam_thickness^2))*(le+me+se);

I.geometry.rays_beam_radius     = 0; % meters
I.geometry.rays_beam_thickness = 0; % meters

I.geometry.stiff_beam_radius     = 0; % meters
I.geometry.stiff_beam_thickness = 0; % meters

% Prints set W/B and materials properties to notepad file
str1 = 'icosahedron_HPC_properties_';
str2 = int2str(hex_radius*2);
str3 = '.txt';
```

```
89  strT = strcat(str1,str2,str3);
90
91  %% FEA Analysis
92  I.step.buckle = 0; % ON(1) / OFF(0), ON disables others
93  % Buckle
94  % Chooses step type
95
96  %% Geometry Calculations
97  % Calculates the icosahedron vertices
98  % Calculates the vertices in cartesian coordinates
99
100 %% Writes variables into Var.py file,which will be read by the main .py file
101
102 %% Runs the Adjusted Script in Abaqus
103 warning('on','all');
104 Rmo = 'noGUI'; % No GUI, analysis runs in the background
105 system(['abaqus cae ',Rmo,'=python2abaqus_',filename,'.py']); % runs the main script
106 %
107 fclose('all');
108 end
```

Listing A.4. HPC Results File

```
1   function [output] = HPC_Results(rb, tb, ts, hex_radius, hex_alt)
2
3   % Gets results from HPC run from odb file (stresses, deflections, weight,
4   % buoyancy)
5
6   I.materials.skin_density  = 970;
7   I.materials.frame_density = 1250;
8
9   % Need to recalculate some info from HPC_Create to get results as the code
10  %is disconnected between two files
11
12  [rho,~,temp,press]=stdatmo(hex_alt*.3048); %ft to meters for the input
13  I.WB.rho   = rho; % air density at SL, kg/m^3, http://en.wikipedia.org/wiki/
        Density_of_air
14  I.WB.g     = 9.81; % acceleration of gravity, m/s^2
15  I.WB.To    = temp; % K, external temp (altitude dependent)
16  I.WB.Ti    = I.WB.To; % K, internal temp (altitude and heat transfer dependent)
```

```
17 I.WB.Po     = press; % Pa, external pressure (altitude dependent)
18
19 % Geometry (icosahedron) code
20
21 % Assume hexakis icosahedron, hollow everything
22 se = (sqrt(15*(85-31*sqrt(5)))/11)*(I.geometry.hexirn/I.geometry.hexir);
23 me = (3*sqrt(15*(65+19*sqrt(5)))/55)*(I.geometry.hexirn/I.geometry.hexir);
24 le = (2*sqrt(15*(5-sqrt(5)))/5)*(I.geometry.hexirn/I.geometry.hexir);
25 s = .5*(se+me+le);
26 ta = sqrt(s*(s-se)*(s-me)*(s-le));
27 hexV = (180*(5+4*sqrt(5))/11)*(I.geometry.hexirn/I.geometry.hexir)^3;
28 I.geometry.initial_volume = hexV;
29 I.geometry.skin_thickness = ts;
30 I.geometry.skin_volume = 120*ta*I.geometry.skin_thickness;
31 I.geometry.frame_beam_radius     = rb; % meters
32 I.geometry.frame_beam_thickness = tb; % input
33 I.geometry.frame_volume = (pi*60*(2*I.geometry.frame_beam_thickness*I.geometry.
       frame_beam_radius...
34     -I.geometry.frame_beam_thickness^2))*(le+me+se);
35
36 I.index = hex_radius*2;
37 index = num2str(I.index);
38 I.filename = ['icosahedron',index]; % I.filename; % .py filename
39 I.scratch_folder = 'Temp Scratch Files'; % used to create the scratch folder and the
       enviroment .env file
40
41 filename3 = 'icosahedron_output_HPC';
42 filename2 = [I.filename,'_output'];
43 filename =    I.filename; %'icosahedron6'; % .py filename
44 job_name_odb = [filename,'-Job.odb'];
45
46 %Static Step Information
47 nonlinear_effects = 'ON';%I.step.nonlinear_effects; % ON or OFF
48 buckle = 0;%I.step.buckle; % ON(1) / OFF(0), ON disables others
49 step_type = 0;%I.step.step_type; % use Riks(1), use General(0)
50 stabilization = 1;%2I.step.stabilization; % strain energy stabilization ON(1) / OFF
       (0), ON w/membrane section
51
52 %% Writes variables into Var.py file,which will be read by the main .py file
53
```

```
%% Runs the Adjusted Script in Abaqus

%% Output

% Nodes coordinates
% Frame Instance
% Skin Instance
% Mesh Details
% Nodes Displacements
% Elements Stresses
% Strain Energy vs. Time

%W/B including Volume Reduction
end
```

Listing A.5. FEA Mediator File

```
% By Adorno-Rodriguez, Ruben
% Edited by Schwemmer, Joseph
% Last updated: Jan 17, 2016
% Function: runs the FEA model of the icosahedron in Abaqus and reads in
% results
function [output, output_abaqus]=icosahedron_fea(I)
%% Input
% Static Step Information
%% Runs the icosahedron_fea_inner(I) function
% Runs the FEA Analysis
O1 = icosahedron_fea_inner(I);
status = O1.system.status; % 0 if succesful, nonzero otherwise
cmdout = O1.system.cmdout; % detailed message

if status == 0 % 0(no error),otherwise(error)
    disp('Analysis completed succesfully!')
    % Reads-in and Saves the FEA outputs
    % Geometry
    output.geometry.vertices    = O1.geometry.vertices;  % vertices
    output.geometry.midpoints   = O1.geometry.midpoints; % edge midpoints
    output.geometry.facecenters = O1.geometry.facecenters; % face centers
    [q, output_abaqus]  = icosahedron_fea_output1(I);
else
```

```
    disp('All the initial increments in the increment vector failed. Function will
        stop')
    % Geometry
    output.geometry.vertices     = O1.geometry.vertices;  % vertices
    output.geometry.midpoints    = O1.geometry.midpoints; % edge midpoints
    output.geometry.facecenters = O1.geometry.facecenters; % face centers

    % Make point infeasible
    output_abaqus.weight = Inf;
    output_abaqus.weightframe = Inf;
    output_abaqus.weightskin = Inf;
    output_abaqus.weightvol = Inf;
    output_abaqus.buoyancy = 1;
    output_abaqus.frame(1).U(1,5) = 0;
    output_abaqus.frame(1).U(1,6) = 0;
    output_abaqus.frame(1).U(1,7) = 0;
    output_abaqus.skin(1).U(1,5) = 0;
    output_abaqus.skin(1).U(1,6) = 0;
    output_abaqus.skin(1).U(1,7) = 0;
    output_abaqus.frame(1).S(1,2) = 10e9;
    output_abaqus.skin(1).S(1,2) = 10e9;
end

end
```

Listing A.6. Output File

```
% By Adorno-Rodriguez, Ruben
% Edited by Schwemmer, Joseph
% Last updated: Jan 15, 2017
% Function: reads in the results from the .dat files
function [output]=icosahedron_fea_output2(I)
%% Reads-in and Saves the FEA outputs
% Nodes coordinates
% Frame Instance
% Skin Instance
% Mesh Details
% Nodes Displacements
% Elements Stresses
f = fopen(['results_',I.filename,'_S.dat']);
```

```
stress = textscan(f,'%f %s %*s %f %f %*[^\n]','HeaderLines',1);
fclose(f);
s  = cell2mat(stress(:,[1  3:end])); % increment, element #, Mises
s1 = s(strcmpi(stress{:,2},{'Frame'}) == 1,:);
s2 = s(strcmpi(stress{:,2},{'Skin'}) == 1,:);

output.frame(1).S = s1(inc(length(inc)) == s1(:,1),2:end); % element #, Mises
output.skin (1).S = s2(inc(length(inc)) == s2(:,1),2:end); % element #, Mises

output.WB = ((I.geometry.skin_volume*I.materials.skin_density+Vframe*I.materials.
    frame_density+I.payload)/...
    ((I.geometry.initial_volume-Vr)*(Po/(R*To))))+((Pi*To)/(Po*Ti));

output.weight = I.geometry.skin_volume*I.materials.skin_density+Vframe*I.materials.
    frame_density+I.payload+(I.geometry.initial_volume-Vr)*(Pi/(R*Ti));
output.weightframe = Vframe*I.materials.frame_density;
output.weightskin = I.geometry.skin_volume*I.materials.skin_density;
output.weightvol = (I.geometry.initial_volume-Vr)*(Pi/(R*Ti));
output.buoyancy = (I.geometry.initial_volume-Vr)*(Po/(R*To));
end
```

Listing A.7. MADS Main File

```
function []=nmads2()

% clear global; %<----

global ncnum;
global set3;
global datapass;
global fevalnum;
global Multi_Run;
global aspire;
global reserv;
%global plotthat;

%-------------------------------
%---What You Wouldn't Know---
%-------------------------------
ncnum=2; %Number of non-linear constraints in the Problem file
```

```
18
19 %---Other Initial Choices---
20 %plotthat=0; %If 1 will plot sub-forms (1 or 2 DVs)
21 set3.stoch=0; %If 1 is stochastic, will use R&S
22 datapass.numobjectives=3; %Number of objectives - change based on problem type
23 set3.pollStrategy='OrthoMADS_n+1'; %MADS_n+1, Standard_n+1, OrthoMADS_n+1,
       OrthoMADSr_n+1, or 2n
24 set3.nFunc=40; %FEval limit for solving any of the searches for the utopia
25 set3.search='LHS'; %LHAMM, LHALTON
26 set3.nPoints=5; %Number of points to use in the SEARCH step
27 set3.search2='None'; %If you want to use a second search
28 secondlim=50; %FEval limit for solving any of the searches for filling gaps (i.e. the
       single objective formulations once the utopia is found)
29 dcrit=0.5; %This number times the Euclidean distance between gap endpoints will
       determine the size of the gap for the termination criteria
30 datapass.sfunc=4; %1 will use normalized single-obj formulations, 2 will use the
       product form (from BiMADS); 3 uses SMOMADS ray; 4 MOMADS
31 scheme=2; %1 here will weight recurring gaps using '+1' in the denominator; 2 will
       double the denominator (faster)
32 datapass.indiffs=ones(1,datapass.numobjectives)*10; %Use 10 bins - change based on
       problem type
33 unreps=1; %Num reps to search for utopia/nadir (could just increase fevals too)
34 nmreps=1; %Num reps to solve problem with these settings
35 repgap=1; %If 2 - use both starting iterates for the gap (o.w. 1)
36 datapass.stx=[0.008; 0.0002; 0.0000005];%col vector ? starting iterate [beam radius,
       beam thickness, skin thickness] in m
37 %aerogel runs
38 % datapass.stx=[0.04; 0.0008; 0.0005];%col vector ? starting iterate for no graphene
       [beam radius, beam thickness, skin thickness] in m
39 % datapass.stx=[600; 600];%col vector ? starting iterate for property edit runs [
       frame properties; skin properties]
40 datapass.stp={};%{?2?}; - Leave empty if not MVP
41 % Options.Term.delta      = 1e-4;          % minimum mesh size
42 % Choices for Mesh Control
43 % Options.delta0          = 1;             % initial mesh size
44 % Options.deltaMax        = 1;             % bound on how coarse the mesh can get
45 % Options.meshRefine      = 0.5;           % mesh refinement factor
46 % Options.meshCoarsen     = 2.0;           % mesh coarsening factor
47 %-------------------------
48 %---END EDIT---------------
```

```
49 %-------------------------------------
50 end
```

Listing A.8. MADS Constraint File

```
1  %**********************************************************************
2  % canoeDW_Omega:  User-supplied function for defining Omega, based on p.
3  %**********************************************************************
4  function [A,l,u] = Problem_Omega(n)
5  % For varying beam size and skin thickness
6  % A = [eye(n);0.02 -1 0;-0.025 1 0];
7  % l = [0.008;0.0002;0.0002;-Inf;-Inf];%beam radius, beam thickness, skin thickness
8  % u = [1;1;1;0;0];%beam radius, beam thickness, skin thickness
9
10 %Using graphene - if using aerogels, the constraints need to be changed
11 %here, ABAQUS_Main must be changed using hard edits, and the beam thickness
12 %value is ignored : recommend removing beam thickness from decision
13 %variable vector entirely
14 A = [eye(n);0.02 -1 0;-0.025 1 0];
15 l = [0.008;0.0002;0.00000000033;-Inf;-Inf];%beam radius, beam thickness, skin
       thickness
16 u = [0.01;0.00025;0.000001;0;0];%beam radius, beam thickness, skin thickness
17
18 %% For varying material properties
19 % A = [eye(n); 1 -1; -1 1];
20 % l = [500;500;-Inf;-Inf];%frame rho, skin rho
21 % u = [2000;2000;10;10];%frame rho, skin rho
22
23 %l, u are col vectors
24 return
```

Listing A.9. MADS Simulation Call File

```
1  function [fx,cx] = Problem(x)
2
3  %If have MVP, need to add p as an input argument, and to Line 15
4
5   global datapass;
6  % Other global variables
7
8   if fweighting~=1
```

```
 numobjectives=datapass.numobjectives;
 end

%Call ABAQUS simulation here !!!!!
%ABAQUS_Main(beam radius, beam thickness, skin thickness, payload in kg, model number
    , radius in inches, altitude in feet);
tic
[output_abaqus] = ABAQUS_Main(x(1), x(2), x(3), 0.1, datapass.incr_num, 24, 1000);
toc
datapass.incr_num = datapass.incr_num + 1;
%using aerogels - consider using only 2 decision variables and making the
%appropriate changes to ABAQUS_Main, Problem_mod, Problem_Omega

% changing material properties
% tic
% [output_abaqus] = ABAQUS_Main_Mat(x(1), x(2), datapass.incr_num, 60, 0);
% toc
% datapass.incr_num = datapass.incr_num + 1;

[f,cx]=Prob_mod(output_abaqus, x);
format long
disp('Design Vector (X):')
disp(x);
% disp(f);
disp('Frame Weight (kg):')
disp(output_abaqus.weightframe);
disp('Skin Weight (kg):')
disp(output_abaqus.weightskin);
% disp(output_abaqus.weightvol);
disp('Buoyancy (kg):')
disp(output_abaqus.buoyancy);
disp('Max Frame Deflection (mm):')
disp(1000*max(sqrt(output_abaqus.frame(end).U(:,5).^2 + output_abaqus.frame(end).U
    (:,6).^2 + output_abaqus.frame(end).U(:,7).^2))); %displacement
disp('Max Skin Deflection (mm):')
disp(1000*max(sqrt(output_abaqus.skin(end).U(:,5).^2 + output_abaqus.skin(end).U(:,6)
    .^2 + output_abaqus.skin(end).U(:,7).^2))); %displacement
% disp('Nonlinear Equations (Frame Stress, Skin Stress, W/B):')
disp('Nonlinear Equations (Frame Stress, Skin Stress):')
disp(cx);
```

```
disp('W/B with payload:')
disp(output_abaqus.weight/output_abaqus.buoyancy);

format


return
```

Listing A.10. MADS Objective Function File

```
function [f,cx] = Prob_mod(output_abaqus, x)

global set3;

%f(1), f(2), ... are the objective functions w/out noise.
% Changing beam/skin geometry
f(1)=output_abaqus.weight/output_abaqus.buoyancy;
f(2)=max(sqrt(output_abaqus.frame(end).U(:,5).^2 + output_abaqus.frame(end).U(:,6).^2
    + output_abaqus.frame(end).U(:,7).^2)); %displacement
f(3)=max(sqrt(output_abaqus.skin(end).U(:,5).^2 + output_abaqus.skin(end).U(:,6).^2 +
    output_abaqus.skin(end).U(:,7).^2)); %displacement

% Changing material properties
% f(1) = -x(1);
% f(2) = -x(2);

%This would be used if you had non-linear constraints.
%change denominator based on material used for skin and frame, the number
%used is the yield stress
cx(1)=(max(output_abaqus.frame(end).S(:,2))/3.8e9)-1;
cx(2)=(max(output_abaqus.skin(end).S(:,2))/50e9 )-1;
%use for material opt
% cx(1)=(max(output_abaqus.frame(end).S(:,2))/3.0e9)-1;
% cx(2)=(max(output_abaqus.skin(end).S(:,2))/3.0e9)-1;
% cx(3)=(output_abaqus.weight/output_abaqus.buoyancy)-1;

end
```

Listing A.11. Input Python Script for HPC Runs

```
# Job
**************************************************
mdb.Job(atTime=None, contactPrint=OFF, description='',echoPrint=OFF,explicitPrecision
    =SINGLE,historyPrint=OFF,
    model=model_name,modelPrint=OFF,multiprocessingMode=DEFAULT,name=job_name,
        nodalOutputPrecision=FULL,
    queue=None,scratch='',type=ANALYSIS,userSubroutine='',waitHours=0, waitMinutes=0)

# Write Input File **************************************************
mdb.jobs[job_name].writeInput()
```

Listing A.12. Output Python Script for HPC Runs

```
# Writes nodes coordinates for each instance in separate files
for name, instance in assembly.instances.items():
    f = open('results_' + str(name) +'_nodes_coordinates_' + str(name1) +'.dat','w')

if buckle == 1:
    # Buckling Eigen Values
    f = open('buckling_eigen_values.dat','w')
    for num in odb.steps[stepname].frames:
        f.write('%s\n'%(num.description))
else:
    # Writes the Displacement for each instance for each Frame
    f = open('results_' + str(name1) +'_U.dat','w')
    f.write('LPF(or increment)          Instance          Node          U1
        U2          U3\n')
    for frame in odb.steps[stepname].frames:
        lpf = frame.frameValue # load factor
        for node in frame.fieldOutputs['U'].values:
            f.write('%f '%(lpf))
            f.write('  %s '%(node.instance.name))
            f.write('  %d '%(node.nodeLabel))
            f.write('  %.12e  %.12e  %.12e\n'%(node.data[0],node.data[1],node.data
                [2]))

    f.close()

    # Writes elements stress S1, S2, S3, Mises, Tresca for each instance for each
```

```
        Frame
    f = open('results_' + str(name1) +'_S.dat','w')
    f.write('LPF(or increment)          Instance          Element          Mises\n')
    for frame in odb.steps[stepname].frames:
        lpf = frame.frameValue # load factor
        for element in frame.fieldOutputs['S'].values:
            f.write('%f    '%(lpf))
            f.write('  %s    '%(element.instance.name))
            f.write('  %d    '%(element.elementLabel))
            f.write('  %.3f\n'%(element.mises))

    f.close()
```

Listing A.13. Output Python Script for Non-HPC Runs

```
# Extract Outputs
**********************************************
# Opens the ODB and establishes the frame
odb=session.openOdb(name=job_name_odb, readOnly=False)
assembly = odb.rootAssembly

# Writes the Displacement for each instance for each Frame
f = open('results_' + str(name1) +'_U.dat','w')
f.write('LPF(or increment)          Instance          Node          U1          U2
            U3\n')
for frame in odb.steps[stepname].frames:
        lpf = frame.frameValue # load factor
        for node in frame.fieldOutputs['U'].values:
                f.write('%f  '%(lpf))
                f.write('  %s  '%(node.instance.name))
                f.write('  %d  '%(node.nodeLabel))
                f.write('  %.12e  %.12e  %.12e\n'%(node.dataDouble[0],node.dataDouble
                    [1],node.dataDouble[2]))

f.close()

odb.close()
```

Appendix B. Code Tutorial

AFIT Specifics

The code does not run on the C Drive, most likely due to write permissions, so the code must be placed on the I Drive or maybe another shared drive. The I Drive is your best bet; it is confirmed to work. Runs take about 20 minutes per iteration, so letting the computer work overnight is almost inevitable.

If you are running into issues and have the python2abaqus_modelname.py script available for the run you are trying to do, try running the code line by line in ABAQUS CAE on the bottom tile of the program, use the >>> button. This process gives better error information than trying to run ABAQUS through MATLAB.

Running the Code

The files that are altered and run are: nmads2.m, ABAQUS_Main.m, ABAQUS_Main_Mat.m, Problem.m, Prob_mod.m, and Problem_Omega.m.

Nmads2.m is the main file that runs the rest of the code. In the nmads2.m file, the following settings are changed based on the problem:

- ncnum=3; Number of non-linear constraints in the Problem file
- datapass.numobjectives=2; Number of objectives - change based on problem type
- set3.nFunc=40; FEval limit for solving any of the searches for the utopia
- set3.nPoints=5; Number of points to use in the SEARCH step
- secondlim=50; FEval limit for solving any of the searches for filling gaps (i.e. the single objective formulations once the utopia is found)
- datapass.indiffs=ones(1,datapass.numobjectives)*10; Use 10 bins - change based on problem type

- datapass.stx =[0.008; 0.0002; 0.0002]; col vector starting iterate [beam radius, beam thickness, skin thickness] in m
- datapass.stx=[400; 400]; col vector starting iterate [frame properties; skin properties]

datapass.indiffs changes the speed that the code converges, where a larger number means the code runs faster because the zone of acceptable solutions is larger. set3.nFunc, set3.nPoints, and secondlim change how many iterations are done. set3.nPoints determines how many search points are used and the other lines indicate how many total runs should be done. Sometimes the code runs more runs than the sum of set3.nFunc, set3.nPoints, and secondlim but the run total will be close to that sum.

ABAQUS_Main is used when the code is being used to optimize the beam radius, beam thickness, and skin thickness for given materials, altitude, and radius. ABAQUS_Main_Mat is used when the material is being optimized. The type of problem being run, optimize structure or material, changes which datapass.stx line is used. The first line is for the optimization of the structure, the second is for optimizing materials. Comment out the line not in use. This setting is the starting value for the optimizer.

Additional changes in ABAQUS_Main_Mat can be made to change the size of the beams and the skin. These changes are made in lines 152, 155, and 156. This may be needed to find a solution that converges in ABAQUS for large radii.

Similarly, many other files have duplicated lines where one line is commented out based on the problem type. In Problem_Omega, the constraints of the problem depend on problem type. The code below shows when the optimization of the structure code is running. L and U are the lower and upper bounds respectively. When using graphene, the skin thickness limits are drastically changed as shown in the last group of constraints.

For varying beam size and skin thickness

- A = [eye(n);0.02 -1 0;-0.025 1 0];
- l = [0.008;0.0002;0.0002;-100;-100]; beam radius, beam thickness, skin thickness
- u = [1;1;1;0;0]; beam radius, beam thickness, skin thickness

For varying material properties

- A = [eye(n); 1 -1; -1 1];
- l = [10;10;-Inf;-Inf]; frame rho, skin rho
- u = [300;300;10;10]; frame rho, skin rho

Using graphene

- A = [eye(n);0.02 -1 0;-0.025 1 0];
- l = [0.008;0.0002;0.00000000033;-Inf;-Inf]; beam radius, beam thickness, skin thickness
- u = [0.01;0.00025;0.000001;0;0]; beam radius, beam thickness, skin thickness

In Problem, the choice of ABAQUS_Main or ABAQUS_Main_Mat is made.

ABAQUS_Main(beam radius in m, beam thickness in m, skin thickness in m, payload in kg, model number, radius in inches, altitude in feet);

- tic
- output_abaqus = ABAQUS_Main(x(1), x(2), x(3), 0.1, datapass.incr_num, 24, 1000);
- toc

- datapass.incr_num = datapass.incr_num + 1;

Changing material properties (frame density (kg/m^3), skin density (kg/m^3), model number, radius in inches, altitude in feet)

- tic
- output_abaqus = ABAQUS_Main_Mat(x(1), x(2), datapass.incr_num, 30, 0);
- toc
- datapass.incr_num = datapass.incr_num + 1;

In Prob_mod, the following code is changed. It shows that in the structure optimization, the objectives are to minimize the weight-to-buoyancy ratio, minimize frame deflection, and minimize skin deflection. The objectives in the material property optimization is to maximize the frame and skin densities. The choice in problem changes datapass.numobjectives in the nmads2 file.

Changing beam/skin geometry

- f(1)=output_abaqus.weight / output_abaqus.buoyancy;
- f(2)=max(sqrt(output_abaqus.frame(end).U(:,5)^2 + output_abaqus.frame(end).U(:,6)^2 + output_abaqus.frame(end).U(:,7)^2)); displacement
- f(3)=max(sqrt(output_abaqus.skin(end).U(:,5)^2 + output_abaqus.skin(end).U(:,6)^2 + output_abaqus.skin(end).U(:,7)^2)); displacement

Changing material properties

- f(1) = -x(1);
- f(2) = -x(2);

Also in the same file, the nonlinear constraints are added. The constraints are to prevent the max stress from exceeding the material yield stress for the frame and the skin. In the material optimization, the weight to buoyancy ratio becomes a constraint instead of an objective. The number of constraints here changes ncnum in the nmads2 file.

Change denominator based on material used for skin and frame, the number used is the yield stress

- cx(1)=(max(output_abaqus.frame(end).S(:,2))/3.8e9)-1;
- cx(2)=(max(output_abaqus.skin(end).S(:,2))/3.0e9)-1;

Use for material opt

- cx(1)=(max(output_abaqus.frame(end).S(:,2))/3.0e9)-1;
- cx(2)=(max(output_abaqus.skin(end).S(:,2))/3.0e9)-1;
- cx(3)=(output_abaqus.weight/output_abaqus.buoyancy)-1;

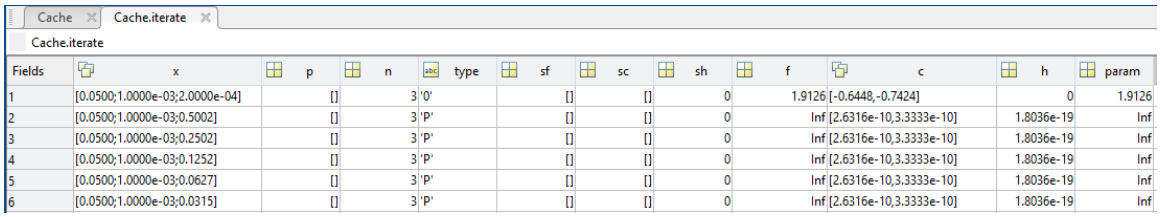
If you are trying to run the code with solid tubes or wanting to change materials, hard edits need to be made to the ABAQUS_Main file by changing which material is not commented out and altering lines 116 and 117 as required for whatever edit was made. Changing to solid tubes involves editing lines 138-140.

Interpreting Results

When the run is completed, two structure arrays, Multi_MADS and nMADS_Results, should be in the workspace. These arrays have some information, but the majority of the information from the run is in the Problem_Cache.mat file. This file has a row for each iteration and gives the design vector, nonlinear constraint evaluations, and objective evaluations. By running the same code over, starting at a different initial point, addition rows of data will be added to the Problem_Cache.mat file without

removing previous work. The Problem_Cache.mat file must be deleted or moved out of the main directory prior to switching major variables on a run (like optimization type, radius of the structure, altitude, essentially anything hard coded).

The most useful data file for looking at results is Problem_Cache.mat. This file will be generated after the initial search for utopia, so after the number of runs is greater than set3.nFunc + set3.nPoints.The information for the run is put in an array called iterate, as shown in the figure below. The first column is the design vector, which contains all of the variables that the program changes. In the hexakis problem, the x vector is the beam radius, beam thickness, and skin thickness. The type column shows if the iteration is an initial point (0), a search point (S), or a poll point (P). The columns f and param show the objective values of the formulation and will be identical if there is just one objective. Param will have all of the objective values if multiple objectives are used, while the value if f will not show the multiple objective values. The column c shows the value of any nonlinear constraints, where answers less than zero are feasible.



Cache | Cache.iterate

Cache.iterate

Fields	x	p	n	type	sf	sc	sh	f	c	h	param
1	[0.0500;1.0000e-03;2.0000e-04]	[]	3	'0'	[]	[]	0	1.9126	[-0.6448,-0.7424]	0	1.9126
2	[0.0500;1.0000e-03;0.5002]	[]	3	'P'	[]	[]	0	Inf	[2.6316e-10,3.3333e-10]	1.8036e-19	Inf
3	[0.0500;1.0000e-03;0.2502]	[]	3	'P'	[]	[]	0	Inf	[2.6316e-10,3.3333e-10]	1.8036e-19	Inf
4	[0.0500;1.0000e-03;0.1252]	[]	3	'P'	[]	[]	0	Inf	[2.6316e-10,3.3333e-10]	1.8036e-19	Inf
5	[0.0500;1.0000e-03;0.0627]	[]	3	'P'	[]	[]	0	Inf	[2.6316e-10,3.3333e-10]	1.8036e-19	Inf
6	[0.0500;1.0000e-03;0.0315]	[]	3	'P'	[]	[]	0	Inf	[2.6316e-10,3.3333e-10]	1.8036e-19	Inf

Additional results are printed to the command window in MATLAB, giving information on the design vector, deflections, stresses, weights, and buoyancy. This data should be copied from the command file and put into a notepad document, as some of this data is not saved for each run. Individual notepad documents with the materials properties, radius, beam geometry, and skin thickness are created for each iteration. You can use these to see how the optimization progressed.

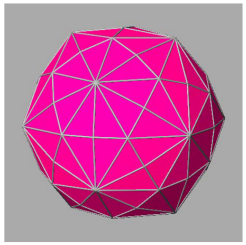
Optimal Design of a Hexakis Icosahedron Vacuum Based Lighter than Air Vehicle

2d Lt Joseph Schwemmer, Advisors: James Chrissis, Ph.D., Anthony Palazotto, Ph.D., Reader: Carl Parson, Ph.D.

Motivation

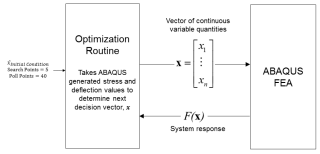
- Due to the rising cost and scarcity of helium, new methods to ensure buoyancy for lighter-than-air vehicles (LTAVs) are being sought
- An internal vacuum can be used to displace air and create buoyancy, but the vacuum creates internal loads on the structure
- Finite-element analysis (FEA) and direct-search methods are employed, providing an optimal design under various regimes
- ABAQUS ® is used as a FEA modeler, and mesh-adaptive direct search (MADS) is the optimization procedure

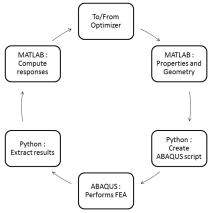
Problem Description



- Objectives:
  - Optimize structure to minimize mass and deflection
  - Reduce diameter from 20 feet (6.096 meters) to a goal diameter of 31 inches (0.7874 meters)
- Considerations:
  - Multiple objectives and constraints
  - No derivative information
  - No closed form objective
  - Computationally expensive
  - altitude $\propto \frac{\text{structure size}}{\text{payload}}$
- Assumptions:
  - Forces on the structure are assumed to be sea level pressure
  - Only vacuum induced loads are examined

Methodology





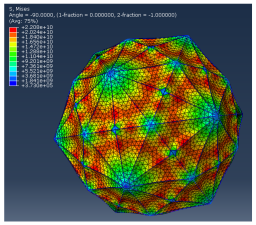
- Static Nonlinear Analysis used in ABAQUS ®
- Boundary conditions to force symmetry, similar to floating condition
- Weight-to-Buoyancy ratio calculated using

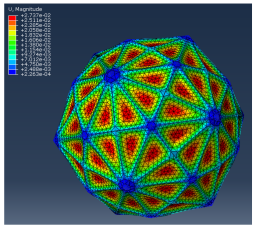
$$\frac{W}{B} = \frac{V_s\rho_s + V_f\rho_f + (V_i - V_r)\rho_{\text{air, i}}}{(V_i - V_r)\rho_{\text{air, o}}}$$

- C-ratio (ratio of beam thickness to beam radius) must be between 0.02 and 0.025

Results and Conclusions

- One type of problem formulation to optimize frame and membrane geometry, another to determine material densities required for given vehicle size
- Two feasible designs found:
  - 15 feet (4.572 meters) diameter with Carbon Nanotube frame and Spectra membrane
    - Weight-to-Buoyancy of 0.9907
    - Beam radius of 30.2 millimeters, beam thickness of 0.678 millimeters, and skin thickness of 0.255 millimeters
    - Maximum altitude of 310 feet (95 meters)
    - Maximum payload of 400 grams
  - 4 feet (1.2192 meters) diameter with Carbon Nanotube frame and Graphene membrane (Shown in Figures)
    - Weight-to-Buoyancy of 0.7654
    - Beam radius is 8 millimeters with a beam thickness of 0.2 millimeters; the skin thickness is 500 nanometers
    - Maximum altitude of 6900 feet (2100 meters)
    - Maximum payload of 200 grams





Direct Search Algorithm

- Initialization:
  - Choose $x_0, \alpha_0 > 0, 0 < \beta_1 \leq \beta_2 < 1$, and $\gamma \geq 1$
  - Let D be a set of positive spanning sets
- Search:
  - Try to compute a point with $f(x) < f(x_k)$ by evaluating the function f at a finite number of points
  - If such a point is found, then set $x_{k+1} = x$, declare the iteration and the search step successful, and skip the poll step
- Poll:
  - Choose a positive basis D_k from the set D
  - Order the poll step $P_k = \{x_k + \alpha_k d : d \in D_k\}$ and evaluate f at the poll points following the chosen order
  - If a poll point is found such that $f(x_k + \alpha_k d_k) < f(x_k)$, then stop polling, set $x_{k+1} = x_k + \alpha_k d_k$, and declare the iteration and the poll step successful
  - Otherwise, declare the iteration (and the poll step) unsuccessful and set $x_{k+1} = x_k$
- Mesh Parameter Update:
  - If the iteration was successful, then maintain or increase the step size parameter: $\alpha_{k+1} \in [\alpha_k, \gamma\alpha_k]$
  - Otherwise, decrease the step size parameter: $\alpha_{k+1} \in [\beta_1\alpha_k, \beta_2\alpha_k]$

Future Research

- Frame material changes
- Formulation changes for optimization
- Aerodynamic analysis
- Design space sampling
- Altering MADS parameters

Contact Information

James Chrissis, Ph.D.
Department of Operational Sciences, AFIT
Anthony Palazotto, Ph.D.
Department of Aeronautics and Astronautics, AFIT

Bibliography

1. B. C. Cranston, *Conceptual Design, Structural Analysis, and Design Space Exploration of a Vacuum Lighter Than Air Vehicle*. Ph.D. dissertation, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433-7765, Mar 2016.

2. R. J. Vanderbei, "Structural design," Statistics and Operations Research SOR-96-08, Princeton University, School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton, New Jersey 08544, Jan 1996.

3. F. H. Gern, "Finite element based HWB centerbody structural optimization and weight prediction," vol. 53 of *Structures, Structural Dynamics and Materials Conference*, (Honolulu, Hawaii), pp. 1–14, AIAA/ASME/ASCE/AHS/ASC, AIAA, Apr 2012.

4. C. R. Parson, J. W. Chrissis, A. N. Palazotto, and R. P. O'Hara, "Direct search optimization of a flapping micro air vehicle wing using FEA characterization of the manduca sexta forewing," in *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 54, (Boston, Massachusetts), Apr 2013.

5. C. Audet and J. E. Dennis Jr., "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, pp. 188–217, May 2006.

6. D. A. Shea and D. Morgan, "The helium-3 shortage: Supply, demand, and options for congress." Congressional Research Service, Dec 2010.

7. 113th Congress, "Helium stewardship act of 2013." Public Law 113-40, Oct 2013.

8. F. Lana-Terzi, T. O. Hubbard, and J. H. Ledeboer, *The Aerial Ship*. Aeronautical Society of Great Britain, 1910.

9. A. Akhmeteli and A. Gavrilin, "Layered shell vacuum balloons," Feb 2006. US Patent App. 11/127,613.

10. T. T. Metlen, "Design of a lighter than air vehicle that achieves positive buoyancy in air using a vehicle," MS thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433-7765, Jun 2012.

11. A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization, 2009.

12. S. N. Patnaik, J. D. Guptill, and L. Berke, "Merits and limitations of optimality criteria method for structural optimization," Technical Paper 3373, NASA Lewis Research Center, Cleveland, Ohio 44135, Nov 1993.

13. T. R. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–395, Mar 2004.

14. E. J. Cramer, J. Dennis Jr., P. D. Frank, R. M. Lewis, and G. R. Shubin, "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, vol. 4, pp. 754–776, Nov 1994.

15. W. L. Winston, *Operations Research: Applications and Algorithms*. Duxbury Press, fourth ed., Jul 2003.

16. F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*. McGraw-Hill Education, tenth ed., Jan 2014.

17. M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, third ed., May 2006.

18. P. Bertsekas, Dimitri, *Nonlinear Programming*. Athena Scientific, third ed., Jun 2016.

19. S. S. Rao, *Engineering Optimization: Theory and Practice*. Wiley-Interscience, third ed., Feb 1996.

20. J. Arora, *Introduction to Optimum Design*. Academic Press, third ed., Aug 2011.

21. D. H. Wolpert and W. G. Macready, "No free lunch theorems for search," Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1399 Hyde Park Road, Sante Fe, NM 87501, Feb 1996.

22. K. Schittkowski, C. Zillober, and R. Zotemantel, "NLPQL: A FORTRAN subroutine solving constrained nonlinear programming problems," *Annals of Operations Research*, vol. 5, pp. 485–500, 1985.

23. K. Schittkowski, C. Zillober, and R. Zotemantel, "Numerical comparison of nonlinear programming algorithms for structural optimization," *Structural Optimization*, vol. 7, pp. 1–19, 1994.

24. M. A. Abramson, "Application of sequential quadratic programming to large-scale structural design problems," MS thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433-7765, Apr 1994.

25. M. A. Abramson and J. W. Chrissis, "Sequential quadratic programming and the ASTROS structural optimization system," *Structural Optimization*, vol. 15, no. 1, pp. 24–32, 1998.

26. T. A. Sriver, "The application of sequential convex programming to large-scale structural optimization problems," MS thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433-7765, Apr 1998.

27. C. Camp, S. Pezeshk, and G. Cao, "Optimized design of two-dimensional structures using a genetic algorithm," *Journal of Structural Engineering*, vol. 124, pp. 551–559, May 1998.

28. I. Kroo, "VKI lecture series on optimization methods & tools for multicriteria/-multidisciplinary design," Nov 2004.

29. R. C. McEachin, "An investigation of simulated annealing applied to structural optimization problems," MS thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433-7765, Mar 1994.

30. R. Hooke and T. A. Jeeves, ""Direct search" solution of numerical and statistical problems," *Journal of the ACM (JACM)*, vol. 8, pp. 212–229, Apr 1961.

31. C. Audet and J. E. Dennis Jr., "Analysis of generalized pattern searches," *SIAM Journal on Optimization*, vol. 13, pp. 889–903, May 2003.

32. F. H. Clarke, "Nonsmooth analysis and optimization," in *Proceedings of the International Congress of Mathematicians*, (Helsinki), 1978.

33. T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, pp. 385–482, Aug 2003.

34. M. A. Abramson and C. Audet, "Convergence of mesh adaptive direct search to second-order stationary points," *SIAM Journal of Optimization*, vol. 17, pp. 606–619, May 2006.

35. M. A. Abramson, C. Audet, J. W. Chrissis, and J. G. Walston, "Mesh adaptive direct search algorithms for mixed variable optimization," *Optimization Letters*, vol. 3, pp. 35–47, 2009.

36. M. A. Abramson, C. Audet, J. E. Dennis Jr., and S. Le Digabel, "ORTHOMADS: A deterministic MADS instance with orthogonal directions," *SIAM Journal on Optimization*, vol. 20, p. 948966, Jul 2009.

37. C. Audet, G. Savard, and W. Zghal, "Multiobjective optimization through a series of single-objective formulations," *SIAM Journal on Optimization*, vol. 19, p. 188210, Feb 2008.

38. T. A. Sriver, J. W. Chrissis, and M. A. Abramson, "Pattern search ranking and selection algorithms for mixed variable simulation-based optimization," *European Journal of Operational Research*, vol. 198, no. 3, pp. 878–890, 2009.

39. T. J. Paciencia, "Multi-objective optimization of mixed variable, stochastic systems using single-objective formulations," MS thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433-7765, Mar 2008.

40. M. D. Payne, J. W. Chrissis, E. A. Pohl, R. D. W. Bowersox, M. R. Gruber, and R. P. Fuller, "Optimizing scramjet fuel injection array design," in *Joint Propulsion Conference*, 35, AIAA/ASME/SAE/ASEE, AIAA, Jun 1999.

41. Z. Zhao, J. C. Meza, and M. Van Hove, "Using pattern search methods for surface structure determination of nanomaterials," *Journal of Physics: Condensed Matter*, vol. 18, p. 86938706, Sep 2006.

42. R. Adorno-Rodriguez, "Nonlinear structural analysis of an icosahedron and its application to lighter than air vehicles under a vacuum," MS thesis, Air Force Institute of Technology, 2950 Hobson Way, WPAFB, OH 45433-7765, Mar 2014.

43. L. Lin, "Introduction to finite element modeling," Aug 2010.

44. X. Wang, Z. Yong, Q. Li, P. Bradford, W. Liu, D. Tucker, W. Cai, H. Wang, F. Yuan, and Y. Zhu, "Ultrastrong, stiff and multifunctional carbon nanotube composites," *Materials Research Letters*, vol. 1, pp. 19–25, Mar 2013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
23–03–2017	Master's Thesis	Sept 2015 — Mar 2017

4. TITLE AND SUBTITLE

Optimal Design of a Hexakis Icosahedron Vacuum Based Lighter than Air Vehicle

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Schwemmer, Joseph R., 2d Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-MS-17-M-158

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of Scientific Research
875 North Randolph Street, Suite 325, Room 3112,
Arlington, VA., 22203-1768
Mr. James Fillerup
James.fillerup@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A. Approved for Public Release;Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Due to the rising cost and scarcity of helium, new methods to ensure buoyancy for lighter-than-air vehicles (LTAVs) are being sought. One alternative under study uses an internal vacuum to reduce the weight to buoyancy ratio. It's a novel approach; however, the vacuum presents challenges for the vehicle's structure. The structure must have minimum mass while preventing buckling and excess stress throughout the frame and membrane. The structure under analysis is a hexakis icosahedron with a membrane covering. Achieving minimum mass involves optimizing the structure under the loading conditions. Finite-element analysis (FEA) and direct-search methods are employed, providing an optimal design under various regimes. Specifically, ABAQUS ® is used as a FEA modeler, and mesh-adaptive direct search (MADS) is the optimization procedure. The goal of this research is to reduce the diameter of the vehicle using optimization techniques to a goal size of 31 inches (0.7874 meters). The smallest design to date has a diameter of 20 feet (6.096 meters). This research demonstrates the feasibility of two designs, one at 15 feet (4.572 meters) and another at 4 feet (1.2192 meters). The problem formulation includes multiple black-box objectives and constraints. Results for a number of designs are presented and compared.

15. SUBJECT TERMS

LTAV, Hexakis Icosahedron, MADS, Direct Search, Structural Optimization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. J. W. Chrissis, AFIT/ENS
U	U	U	UU	99	19b. TELEPHONE NUMBER *(include area code)* (937) 785-3636, x4606; james.chrissis@afit.edu